Computers & Geosciences 106 (2017) 60–67

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Spatial coding-based approach for partitioning big spatial data in Hadoop

Xiaochuang Yao[a], Mohamed F. Mokbel[b], Louai Alarabi[b], Ahmed Eldawy[c], Jianyu Yang[a], Wenju Yun[d], Lin Li[a], Sijing Ye[e], Dehai Zhu[a,*]

[a] College of Information and Electrical Engineering, China Agricultural University, Beijing 100083 China
[b] Department of Computer Science and Engineering, University of Minnesota, MN 55455 USA
[c] Department of Computer Science and Engineering, University of California Riverside, CA 92521 USA
[d] Land Consolidation and Rehabilitation Center, Ministry of Land and Resources, Beijing 100035 China
[e] Institute of Remote Sensing and Digital Earth Chinese Academy of Sciences, Beijing 100094 China

ARTICLE INFO

Keywords:
Spatial coding-based approach
Big spatial data
Spatial data partitioning
Hadoop

ABSTRACT

Spatial data partitioning (SDP) plays a powerful role in distributed storage and parallel computing for spatial data. However, due to skew distribution of spatial data and varying volume of spatial vector objects, it leads to a significant challenge to ensure both optimal performance of spatial operation and data balance in the cluster. To tackle this problem, we proposed a spatial coding-based approach for partitioning big spatial data in Hadoop. This approach, firstly, compressed the whole big spatial data based on spatial coding matrix to create a sensing information set (SIS), including spatial code, size, count and other information. SIS was then employed to build spatial partitioning matrix, which was used to spilt all spatial objects into different partitions in the cluster finally. Based on our approach, the neighbouring spatial objects can be partitioned into the same block. At the same time, it also can minimize the data skew in Hadoop distributed file system (HDFS). The presented approach with a case study in this paper is compared against random sampling based partitioning, with three measurement standards, namely, the spatial index quality, data skew in HDFS, and range query performance. The experimental results show that our method based on spatial coding technique can improve the query performance of big spatial data, as well as the data balance in HDFS. We implemented and deployed this approach in Hadoop, and it is also able to support efficiently any other distributed big spatial data systems.

1. Introduction

In the era of big data, it has been evolved from a data scarce to a data rich or big data environment in many fields of science (Kitchin, 2014; Miller and Goodchild, 2014), which has caused a number of application systems to employe distributed processing and parallel computing frameworks. Hadoop is one such open-source framework, which has been around since 2007 and proven as an efficient framework for big data analysis in many fields, such as, machine learning (Low et al., 2012), bioinformatics (Gaggero et al., 2008), and graph processing (Avery, 2011).

Unfortunately, for big spatial data, Hadoop is unreliable and inefficient as it is designed ignoring characteristics of spatial dataset essentially (Eldawy and Mokbel, 2013). For example, Hadoop employs default HashPartition (Liu, 2013) to split big data into many child blocks with a fixed block size, which can make good data balance and reduce data skew in Hadoop distributed file system (HDFS). However, as Fig. 1(a) shows, this method will disrupt spatial distribution characteristics between neighbouring objects, which is not beneficial to spatial data processing. To solve this problem, some Hadoop based systems, such as Hadoop-GIS (Aji et al., 2013), and SpatialHadoop (Eldawy and Mokbel, 2015) have been developed. So far, SpatialHadoop (Eldawy et al., 2015), the most advanced distributed GIS system of them, employs space partitioning (grid and quad tree), data partitioning (STR, STR+, and K-d tree), and space filling curve partitioning (*z*-curve and Hilbert curve) to make up for the drawback of defaulted partition method in Hadoop. Based on these spatial data partitioning techniques, big spatial data can be grouped into different partitions simply with their spacial locations. However, due to the unevenly distribution of spatial data and varying volume of spatial objects, as shown in Fig. 1(b), it is likely to cause some thin or oversized data blocks to handle with MapReduce job, as well as high data skew in HDFS.

Moreover, sampling (Eldawy et al., 2015; Aly et al., 2015) is adopted to make a spatial data partitioning schedule for big spatial data. Based on sampling, it can reduce the tasks time and improve efficiency without scanning the entire dataset expensively (Aly et al., 2015). However, the

* Corresponding author.
E-mail addresses: 776558327@qq.com (D. Zhu), xcyao@cau.edu.cn (X. Yao).

http://dx.doi.org/10.1016/j.cageo.2017.05.014
Received 15 December 2016; Received in revised form 13 April 2017; Accepted 28 May 2017
Available online 30 May 2017
0098-3004/

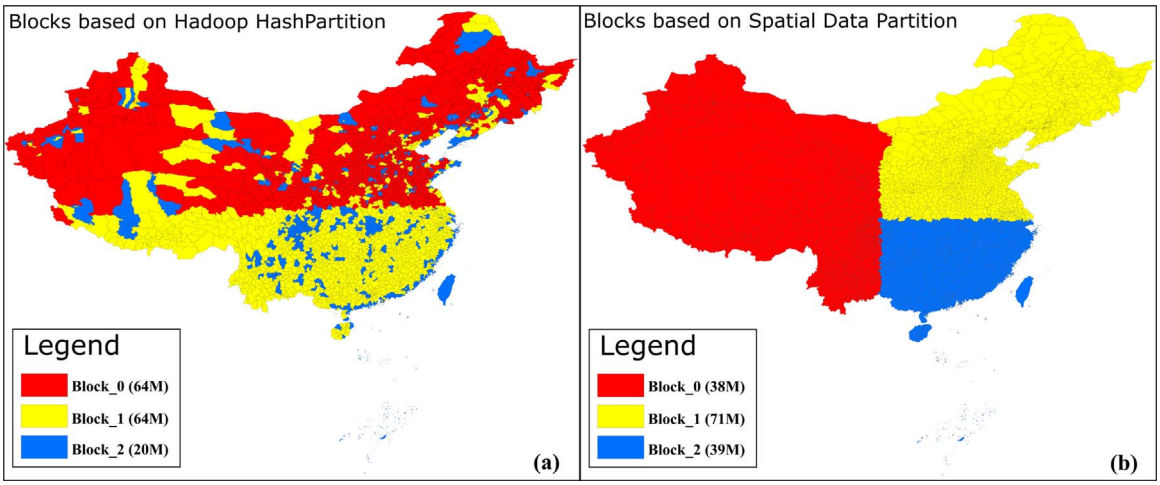

Fig. 1. The blocks based on Hadoop HashPartition (a) and spatial data partition (b).

sampling dataset is controlled and affected by sampling ratio and sampling method (Minasny et al., 2007). It is also quite possibly that some smaller or bigger partitions are produced (Vo et al., 2014). For example, random sampling, which is an unbiased probability method, totally ignores the spatial properties of the original dataset. In addition, due to the randomness of sampling dataset, it cannot guarantee every time we can get the same data partitioning scheme with fixed sampling ratio and the same method. Although spatial sampling can solve these problems with remaining spatial similarity, there still exists the data skew problem as shown in Fig. 1(b).

A good spatial data partition (SDP) strategy should make sure both optimal performance of spatial operation and data balance in the cluster (Wei et al., 2015). This paper presents a spatial coding-based approach (SCA) for partitioning big spatial data efficiently in Hadoop. The method stands for three main steps for partitioning big spatial data. Firstly, we compress the whole dataset, based on spatial coding matrix (SCM), into a sensing information set (SIS), including spatial code, size, count and other information. In this step, users can adopt different spatial codes, such as Hilbert code, Grid code or others. Secondly, we employ SIS to build a spatial partitioning matrix (SPM), which takes the spatial code and block size in HDFS into consideration, and it will compute the partition id for all spatial objects. Finally, the corresponding spatial coding-based data partition will be executed. In addition to running this method as a standalone program, it is also integrated with SpatialHadoop (Eldawy et al., 2015), a scalable MapReduce based framework.

2. Background

2.1. Spatial coding

Spatial coding, which is used for indexing and clustering geographic objects, is a specific implementation method of spatial data structure in a standard database (van Oosterom and Vijlbrief, 1996). By spatial coding, as Fig. 2 shows, each spatial object will be encoded with unique order code to manage (Abel and Smith, 1983; Bajerski, 2008; Bajerski and Kozielski, 2009). Aside from the fact that spatial coding improves the overall manageability of spatial datasets, it also improves spatial data processing in two ways. Firstly, spatial coding can make the spatial data more suitable for computer processing efficiency in one-dimension. Secondly, spatial data will be compressed mostly without loss of spatial location and distribution, avoiding the cost of large computing.

Spatial coding is widely applied in GIS, especially for spatial index (Hadjieleftheriou et al., 2005), such as grid, quad tree and Hilbert-R tree. For distributed and parallel GIS systems, the spatial coding is frequently used for spatial data partitioning. However, to the best of our knowledge, existing methods (Eldawy et al., 2015; Vo et al., 2014) just take more consideration of spatial locations, not other spatial properties based on spatial coding, such as the size of spatial objects, count and so on, which are highly beneficial for data balance in HDFS.

2.2. Spatial data partitioning in Hadoop

Data partitioning plays an powerful role in distributed storage and parallel computing for big data (Scheuermann et al., 1998). Based on data partitioning, big data can be divided into relatively small and independent child blocks, which is a basic and powerful mechanism for improving efficiency of data storage and management systems. In addition to this, the idea, “divide and conquer” from data partitioning also can improve data processing and computing. For example, if the data partitioning is performed effectively, it just needs to scan a few partitions instead of whole dataset in data retrieval or query operation.

Spatial data partitioning (SDP), because of the skew distribution of spatial data (Wei et al., 2015) and varying volume of spatial objects, is significantly differentiated from the database management system (DBMS), which simply adopts horizontal and vertical partitioning techniques (Agrawal et al., 2004). Existing systems (Aji et al., 2013; Eldawy et al., 2015) employ many spatial partitioning methods to bridge this gap. However, to make sure both optimal performance of spatial operation and data balance in Hadoop cluster, we should take into consideration all following points:

(1) Spatial Objects. Spatial objects are the smallest unit of spatial data partitioning. Therefore, in the division process, any spatial object should be not split.
(2) Spatial Location. Usually, the geometric approximation, such as center, minimum bounding rectangle (MBR), convex hull, etc., are used to represent the complete two-dimensional geometry (Polyline and Polygon).
(3) Spatial Distribution. Spatial objects always tend to be spatially correlated and skew distribution. Therefore, spatially adjacent objects should be partitioned into the same blocks as much as possible.
(4) Object Volume. Object volume is to describe the size by bytes in physical storage level. This is an extremely important factor for data balance, yet, it is almost ignored in existing spatial data partitioning methods.
(5) Block Size. Block Size in HDFS is an another standard for data partitioning, which determines whether the data block will be subdivided or merged.

3. Methodologies

3.1. Spatial coding-based approach

In this paper, we use spatial coding-based approach (SCA) instead of sampling to make the data partitioning schedule. As Fig. 3 shows,

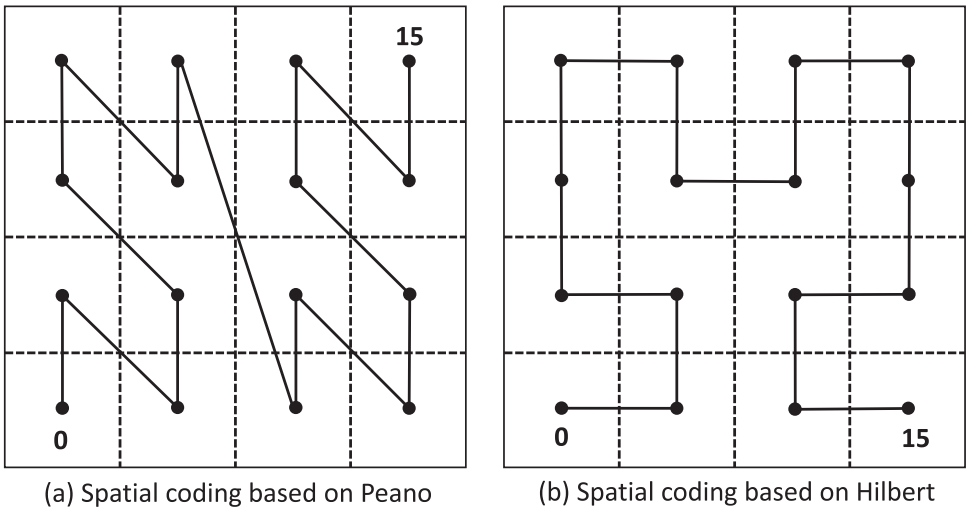

Fig. 2. Spatial coding based on Peano (a) and Hilbert (b).

the original dataset will be compressed with spatial code (van Oosterom and Vijlbrief, 1996). We define a dataset having the same spatial code as one spatial coded block. In this coded block, we will collect and sense spatial properties with location and code, which is good for the spatial data partitioning and makes sure the spatial objects are neighbors in the same coded block. We also will gather and compress other informations, such as size and count, which are benefit to the data balance in HDFS. By considering all influence factors in Section 2.2, the steps for SCA in this paper are as follows:

(1) Computing spatial location. Here, we use the point as spatial location. For the two-dimensional geometry, such as polyline and polygon, we will use their center points instead of the spatial objects themselves.
(2) Defining spatial coding matrix (SCM). Given a big spatial data, we divide the space into grid cells with spatial coding. We define the spatial coding values as a spatial coding matrix (SCM), say A, and define grid cells as spatial coded blocks, as Fig. 3 shows. According to spatial code, we can quickly identify the unique order code for each spatial object in the dataset.
(3) Computing sensing information set (SIS). For each spatial coded block in SCM, it will be compressed spatially to get a corresponding sensing information set (SIS). As Fig. 3 shows, the SIS will be contained with spatial code, spatial location, sum size and count of the spatial coded blocks and others. SIS are the final results after compressing the whole big spatial data.

    Here, for the larger spatial coded blocks than default size, we also need to collect a sub-split set, which is used to divide the large coded blocks again. As Fig. 4 shows, if the coded blocks like (a), firstly, we will compute its average size $\overline{V}$, then order the objects in this block by x coordinates, finally, get a x collections, $\{x_0, x_1, x_2, \cdots\ldots, x_t\}$, according to a certain interval x, and $x = \frac{BlockSize}{\overline{V}}$. If the coded blocks like (b), y coordinates will be replaced.
(4) Computing spatial partitioning matrix (SPM). We use the SIS to estimate and build spatial partitioning scheme, namely, spatial partitioning matrix (SPM). In this step, as the Fig. 5 shows, we need to compare the size of the spatial coded block with the fixed block size in HDFS. If the size is smaller, the coded block will have the same id number with its neighbouring coded blocks, until their sum size is similar to the fixed block size in HDFS. Otherwise, it will be sub-split into more blocks according to the sub-split set in sensing information set (SIS), and the remaining small fragments also will be handled with neighbouring blocks.

3.2. SCA based data partitioning

On the basic of spatial coding-based approach (SCA), we can get the spatial partitioning matrix (SPM) for the whole big spatial data. Next, we just do the spatial data partitioning as the following two steps:

(1) Spatial data partitioning. SPM is the data partitioning scheme. Here, we traverse each data record and find its corresponding block id for every spatial object by matching the spatial code, and then write the object into the data block in HDFS.
(2) Allocating the data blocks. In the SPM, there is the block id for every coded block with spatial code, therefore, it is very easy to finish this step for the whole spatial data with MapReduce efficiently.

3.3. Architecture of spatial partitioning based on SCA

In this section, we present the architecture of spatial partitioning based on spatial coding-based approach (SCA). Fig. 6 summarizes the six steps, which can be done by two MapReduce jobs in Section 3.4. Compared to the sampling based methods, our approach takes full advantage of spatial coding in the 2 step to collect basic information. Based on the spatial coding matrix (SCM), in the 3 step, we give full consideration to spatial relationship of adjacent objects, the size of spatial object and count of spatial coded blocks to make a better schedule for data partitioning. In the 4 step, we will get a data partitioning schedule and in the 5 and 6 step, big spatial data will be partitioned into data blocks and distributed into nodes in the Hadoop cluster spatially.

3.4. Case study: HCA for partitioning big spatial data

In this paper, the Hilbert coding-based approach (HCA) is developed for spatial data partitioning over MapReduce. Hilbert curve is a classic space-filling curve, constructed by the German mathematician Hilbert (Hilbert et al., 1981). Due to its excellent spatial clustering performance for the two-dimensional objects (Abel and Mark, 1990), Hilbert spatial-filling curve is commonly used in spatial data processing. In Table 1, we define some symbols for the pseudo codes in details.

Based on above symbols, our algorithm can be implemented with two MapReduce jobs. One will realize the first three steps in Fig. 6. And the rest steps will be achieved by the second MapReduce job. According to the structure of Hilbert curve, n order Hilbert curve will have $2^n \times 2^n$

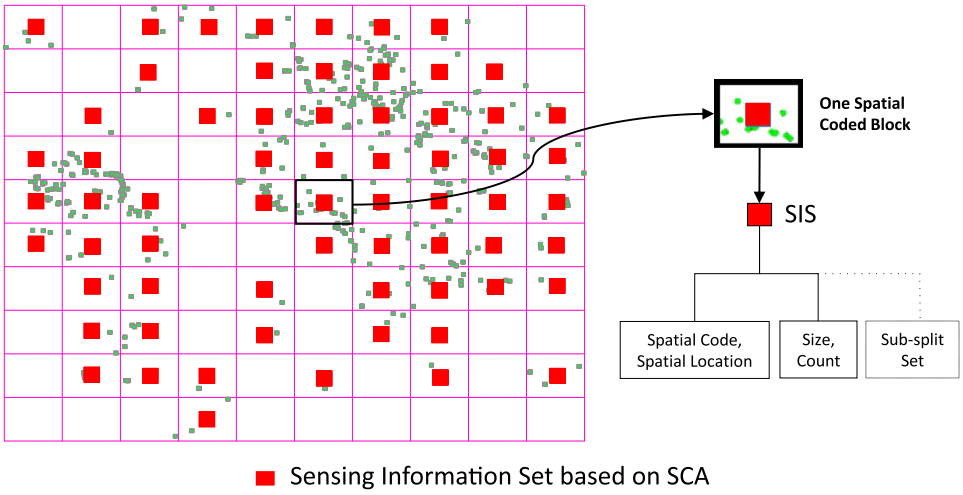

Fig. 3. Sensing information set (SIS) based on SCA.

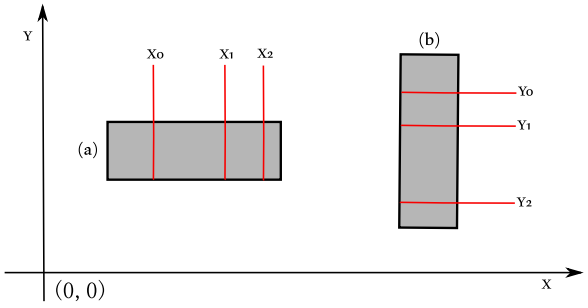

Fig. 4. Sub-split set for bigger spatial coded blocks.

grid cells. Here, we suggest the value of initial order $M_0 = \left\lceil \log_2 \frac{Volume}{BlockSize} \right\rceil$.

In the first MapReduce algorithm, as Table 2 shows, map phase will get the spatial basic information about every object. And reduce phase will compute the sensing information set (SIS) for every Hilbert coded block. Meanwhile it will also get the sub-split set for the large Hilbert coded blocks. Firstly, we calculate the average volume of objects $\overline{V}$, according to the formula: $\overline{V} = \frac{\sum_{i=0}^{t} size_i}{t}$. Then, all spatial objects in this block will be ordered by X or Y coordinates as Fig. 4 shows. We can get a certain interval number by this formula: $x = \frac{BlockSize}{\overline{V}}$. Based on x, we can get a series of X (or Y) coordinates set, which is further used to split the large Hilbert coded blocks again.

In the second MapReduce algorithm, as Table 3 shows, map phase will compute the spatial partitioning matrix (SPM), and reduce phase will partition the full dataset and distribute all the blocks into nodes in the cluster. In the first phase, we use the sensing information set (SIS) to calculate the data block id for every Hilbert coded block, by merging the small blocks and sub-split the large one with a threshold ρ. Next, according to SPM, the whole big spatial data will be partitioned spatially, and neighbouring objects will be wrote into one block. Finally, all of the blocks will be distributed into nodes in the cluster spatially.

4. Evaluation and discussion

In this section, we present the detailed experiments and results with five real spatial datasets in Table 4. And the Hilbert coding-based approach in this paper is compared against random sampling based partitioning, with three measurement standards, namely, the spatial

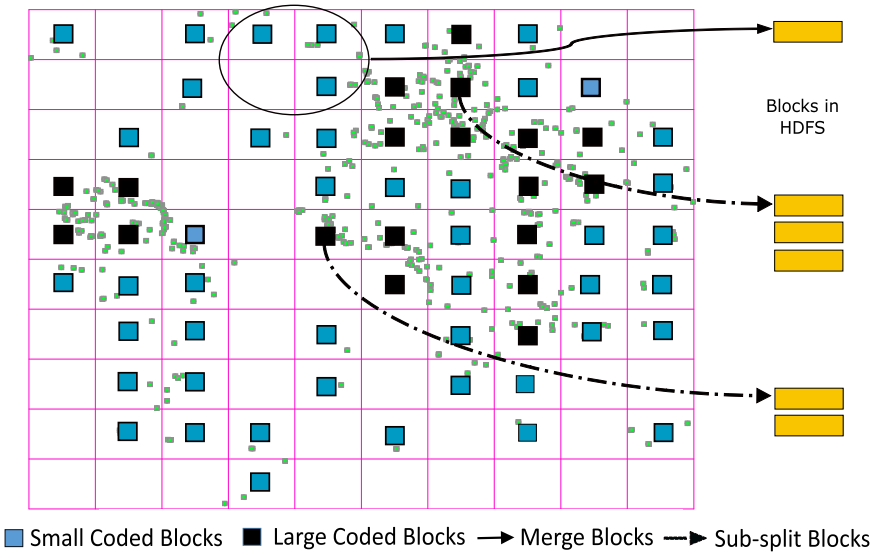

Fig. 5. Spatial partitioning matrix (SPM) based on SCA.

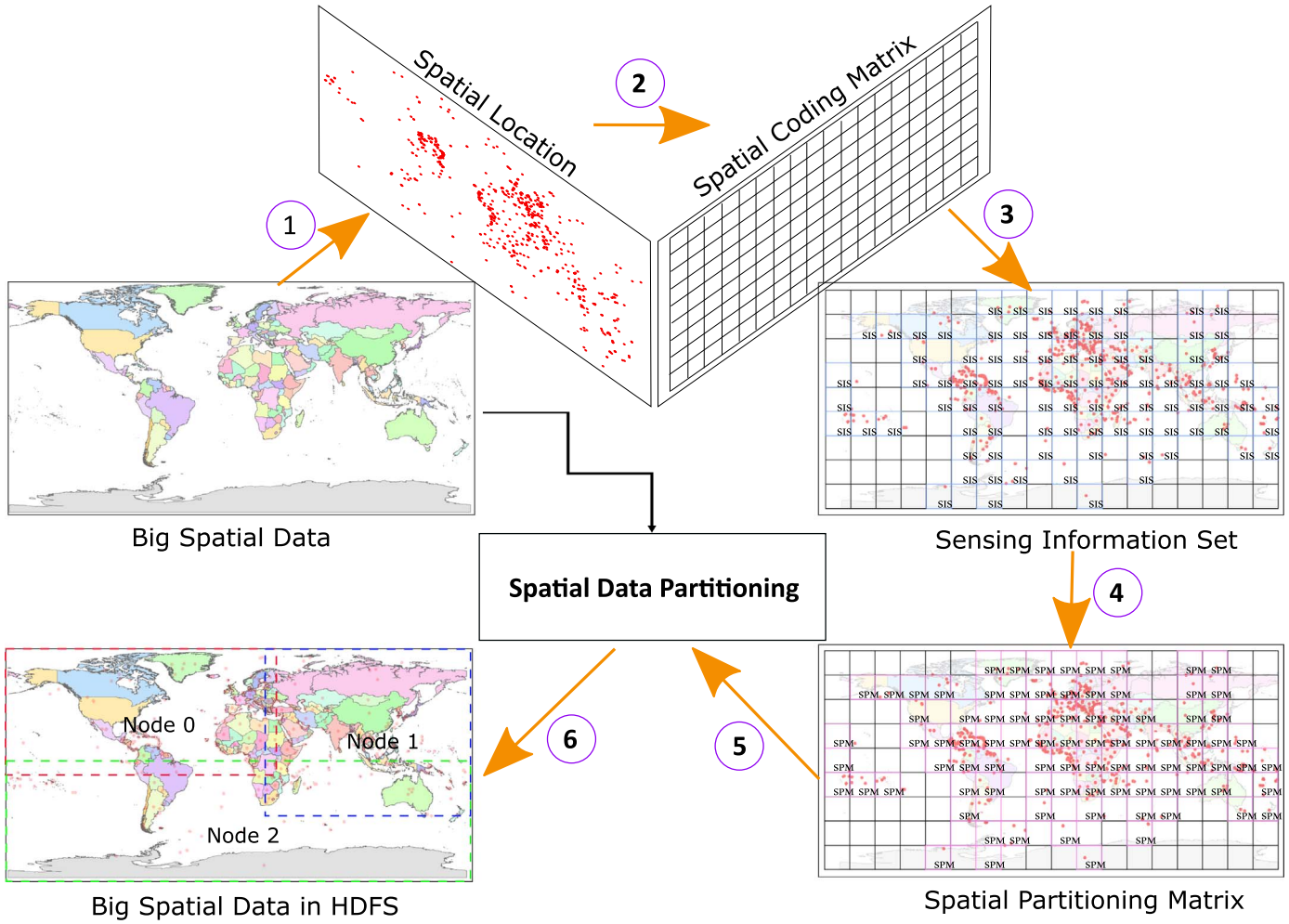

Fig. 6. The architecture of spatial partitioning based on SCA.

Table 1
The symbols for HCA based spatial partitioning over MapReduce.

Symbols	Definition
Volume	Size of big spatial data
M_0	Initial order number of Hilbert curve
$\overline{V}$	Average volume of objects in the same Hilbert coded block
x	Certain interval number for sub-split set
size	Size of spatial object
t	Number of objects in one Hilbert coded block
i	Order position of spatial object in sup-split set
BlockSize	Size of fixed blocks in HDFS
ρ, ρ_{min}, ρ_{max}	a threshold for the blocksize in HDFS
N	Number of nodes in cluster

index quality, data skew in Hadoop distributed file system (HDFS), and query performance.

4.1. Experimental design

Hadoop cluster: we used a hadoop cluster including 8 computer nodes to evaluate the effectiveness and efficiency of our proposed approach. Each PC node is equipped with 4*16 GB RDIMM and runs the Ubuntu 14.0 operating system. We adopt Hadoop 1.2.1 and the default block size in HDFS is 64 MB.

Table 2
The first MapReduce algorithm.

First MapReduce Algorithm: Hilbert coding-based approach (HCA)	
1	Input: big spatial data D
2	Start
3	// Map Phase
4	// Get the spatial basic information about every object
5	For (shape: shapes)
6	{
7	cPoint=shape.getCenterPoint(); // Get center point
8	size=shape.getSize(); //Get size
9	hCode=HilbertCode.getOrder(cPoint); // Get Hilbert code
10	}
11	// Reduce Phase
12	// Computing sensing information set (SIS) for every Hilbert coded block
13	For (hCode: hCodes)
14	{
15	$sumSize = \sum_{i=0}^{t} size_i$//Get the sum size of every Hilbert coded block
16	}
17	if(sumSize > BlockSize*ρ_{max})
18	subSplitSet=hcode.getSubSplit(); //Get sub-split set for large coded block
19	else
20	subPartitionSet=0;
21	End

Table 3
The second MapReduce algorithm.

Second MapReduce algorithm: data partitioning	
1	Input: big spatial data D
2	//Start
3	//Map Phase
4	//Computing spatial partitioning matrix (SPM)
5	For (hCode: hCodes)
6	{
7	//Merging small Hilbert coded blocks with neighbors
8	if (subPartitionSet==0)
9	{
10	do{
11	mergerSize=sumSize + sumSize. Next; //Merging size
12	next.blockId=blockId; //Set the same block id for mergered bloks
13	}while (sumSize. Next & & mergerSize < BlockSize*ρ_{min})
14	}
15	//Sub-split large Hilbert coded blocks
16	else{
17	blockId=blockId + i;
18	if (last. subSplit < ρ_{min})
19	blockId=blockId + 1; //Merging the remaining small fragment
20	}
21	}
22	//Reduce Phase
23	//Spatial partitioning and blocks distribution
24	For (shape: shapes)
25	{
26	cPoint=shape.getCenterPoint(); //Get center point
27	hCode=HilbertCode.getOrder(cPoint); //Get Hilbert code
28	blockId=SPM.getBlockId(hcode); //Get the blockId in HDFS for each spatial object
29	write(blockId, shape); //Writing the shape into data blocks
30	}
31	//End

Table 4
The datasets in experiments.

Name	Size	Records	Average record size	Symbol
World counties	2.9 GB	255 K	11.9 kb	D0
Lakes	9.3 GB	10 M	999 bytes	D1
Roads	25 GB	109 M	234 bytes	D2
All ways	59.6 GB	164 M	390 bytes	D3
All objects	92.5 GB	263 M	378 bytes	D4

Spatial Dataset: we use five real spatial datasets in our experiments. As shown in Table 4, the size is form 2.9 GB to 92.5 GB.

4.2. Results and discussion

4.2.1. Spatial index quality

Spatial data partitioning is powerful mechanism for improving efficiency of spatial index directly. Here, the R-tree performance quality is employed to measure the data partition results (Eldawy et al., 2015). For the R-tree performance quality, referring two main parameters (Cary et al., 2009), namely, Area(T) and Overlap(T). Minimizing both Area(T) and Overlap(T) is known to improve the R-tree performance quality, because they increase path pruning abilities of R-tree navigation algorithms (Beckmann et al., 1990).

Figs. 7 and 8 show separately Area(T) and Overlap(T) comparison for real datasets based on random sampling and Hilbert coding-based approach (HCA).

According to the results, it can be found that HCA method for spatial data partitioning presents a better performance than random sampling method in both Area(T) and Overlap(T), which means HCA

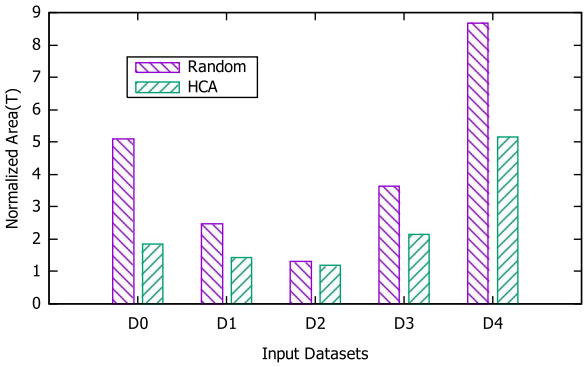

Fig. 7. The area(T) comparison for five real datasets based on random and HCA.

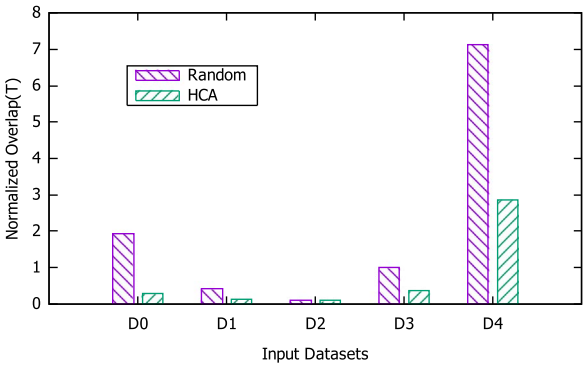

Fig. 8. The Overlap(T) comparison for five real datasets based on random and HCA.

gives a better schedule for big spatial data. It is mainly due to that spatial coding-based approach is taking the spatial distribution characteristics of the original datasets into consideration when it is making a data portioning schedule, comparing to the random sampling. And in the partitioning step, it puts the neighbouring objects into one data block as much as possible according to their spatial codes.

4.2.2. Data skew in HDFS

For the data skew in Hadoop distributed file system (HDFS), we compute the statistical information, such as max/min/average/standard deviation and coefficient of variation, of data blocks in HDFS. In this paper, the standard deviation (SD) and coefficient of variation (CV) are adopted to measure the data block skewness across partitions in HDFS. Higher SD indicates the data values have a wider range and smaller CV implies better load balancing or data skew in HDFS.

Table 5 shows the statistical results of data blocks in HDFS for five real datasets based on random sampling and Hilbert coding-based approach (HCA). From the statistical results, we can find random

Table 5
The statistical results of data blocks in HDFS based on random and HCA.

Dataset	Method	Max/mb	Min/mb	Avg/mb	SD	CV
D0	Random	227.030	2.177	52.019	49.966	0.961
	HCA	61.736	35.230	51.106	3.094	0.061
D1	Random	119.582	16.737	53.247	22.805	0.428
	HCA	68.492	16.347	54.135	4.015	0.074
D2	Random	166.360	22.667	53.016	17.012	0.321
	HCA	87.031	27.060	59.290	10.003	0.169
D3	Random	144.315	19.359	53.172	20.974	0.394
	HCA	106.680	33.835	66.482	9.902	0.149
D4	Random	247.110	12.854	53.167	21.619	0.407
	HCA	101.004	48.123	64.168	8.527	0.133

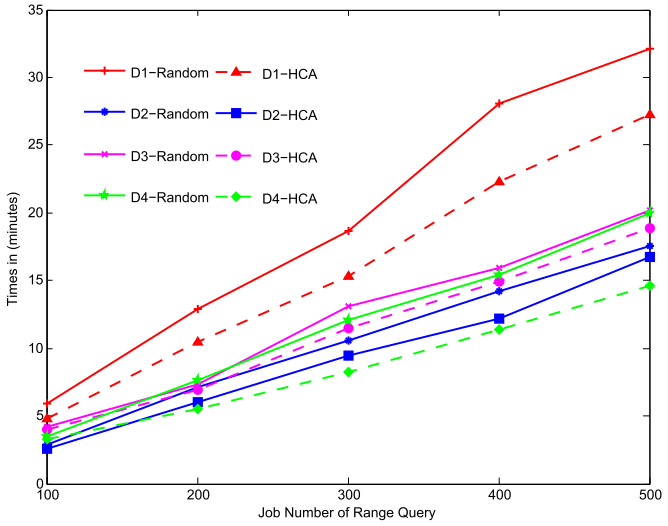

Fig. 9. Query performance of the contrast between different job numbers.

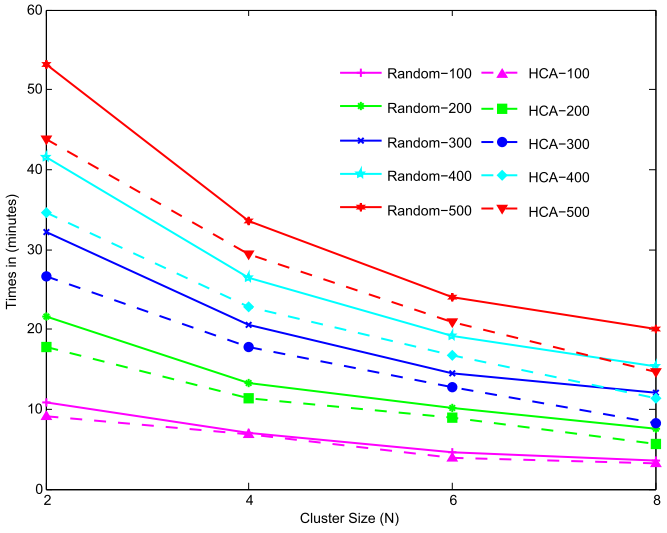

Fig. 10. Query performance of the contrast between different cluster sizes.

sampling will lead to heavily data skew of data blocks in HDFS. For the max size of the data blocks, random is about twice as large as HCA, and for the SD, the random is about 5 times to HCA. All CVs based on the HCA for the five real datasets are smaller than random sampling. There are two reasons that make HCA as a good data balance in this paper. One is that we take more information, such as size, count, etc., about the original dataset into data partitioning schedule. The other one is we take the default block size in HDFS into consideration when the dataset is divided. Based on these, it can make sure the size of the data block is as close as possible to the default size in HDFS.

4.2.3. Query performance

Based on the R-tree, we perform range query to test the query performance of the partitions. In each experiment, to avoid randomness of single range query, we measure the query time of the cluster to answer a batch of queries of jobs by generating some [1*1] grids randomly. And we also test the range query time for the biggest dataset in different number of nodes.

Fig. 9 shows the query performance of the contrast between different job numbers for four real datasets based on random sampling and HCA. In this test, 8 data nodes are used and the job number is from 100 to 500. From the results, we can find that the range query time is increasing with the increase of the number of job tasks. Although we are not able to compare the difference between different datasets because of their spatial distribution, it can be shown HCA based spatial partition is better than random sampling for all datasets in this paper.

In the second test, we choose the largest dataset, all objects with 92.5 GB, and submit 100–500 job tasks based on 2–8 nodes. As a result, the overall execution time on various numbers of job tasks and nodes are shown in Fig. 10. For the same number of job tasks, the range query time will be shorter and shorter with the increase of cluster nodes. It is clearly shown that the results of HCA are still better, and have advantages than random sampling for all datasets.

The improvement of the query performance of the spatial data is due to two reasons in this paper. The first is that the neighbouring spatial objects are split into the same blocks which is exceedingly conducive to spatial processing. The second is the data blocks distribution in HDFS is more balanced, which can avoid time consuming for some partitions containing a lot of spatial objects. Based on these factors, such as spatial objects, location, and others, all of them are fully considered when it makes the spatial data partitioning schedule in spatial coding-based approach (SCA) for partitioning, therefore, we can conclude that our proposed algorithm has an excellent query performance for big spatial data.

5. Related work

Distributed frameworks and parallel computing provide an ideal and practical solution for processing big spatial data (Hawick et al., 2003). However, spatial dataset parallelism, namely, spatial data partitioning (SDP), is particularly challenging for both optimal performance of spatial operation and data balance in the cluster. Spatial dataset can be partitioned into child groups based on their location (latitude and/or longitude), spatial grid cells (Ma and Zhang, 2007), or space-filling curves (Hungershöfer and Wierum, 2002; Meng et al., 2007). These methods are able to simply and quickly partition big spatial data spatially, however, they give no consideration to data balance in cluster. Furthermore, early spatial data partitioning algorithms are to simply divide large dataset into different child groups, which are then processed by different processors (Ye et al., 2011). Despite they can achieve the preliminary purpose of the data partitioning, when faced with the large dataset, there also have some challenges as following. Firstly, almost all of the dataset will be involved in the development of data partitioning strategy, without sampling or compressing. Secondly, the dataset in one node is still a complete data block, not blocks. Last but not least, the algorithms themselves are not realized by parallelizing.

Spatial dataset, itself, tends to be heavily data skew, not only because of the unevenly distribution of spatial dataset (Wei et al., 2015; Zhao et al., 2016), but also due to their varying sizes. To address above problems, in the past few years there has been significant progress in the area of the parallel and distributed GIS systems, such as Eagle-eyed elephant (Eltabakh et al., 2013), SpatialHadoop (Eldawy and Mokbel, 2013), Hadoop-GIS (Aji et al., 2013) and Kangaroo (Aly et al., 2016). However, to the best of our knowledge, the state-of-the-art parallel algorithms have not solve the problems well. For instance, the Eagle-eyed elephant was proposed to avoid accesses of data splits. But, it considers only one-dimensional spatial data. SpatialHadoop (Eldawy and Mokbel, 2015) provided more comprehensive and basic techniques for partitioning big spatial data based on random sampling (Eldawy et al., 2015). However, it can produce some oversized or thin blocks due to over/under sampling. For the partitioning results, SATO (Vo et al., 2014) in Hadoop-GIS system, adopted post re-partitioning to optimize data balance. But, it has to re-scan the data to collect basic statistics costly. AQWA (Aly et al., 2015) in Kangaroo system employs a K-d tree based algorithm and balances workload by repartitioning according to the queries. However, it considers only large data blocks based spatial gird as a query operation is executed. Essentially, it also has drawback in data balance.

6. Conclusions

Skew distribution of spatial dataset and varying volume of spatial objects pose a big challenge for spatial data partitioning in distributed GIS systems. We presented a new approach, spatial coding-based approach (SCA), to optimize spatial data partitioning in our research. Based on our algorithm, the whole big spatial data was compressed into a sensing information set (SIS), which took more information about spatial dataset into consideration. And then SIS was employed to build spatial partitioning matrix (SPM), which was used to partition big spatial data finally. In this paper, a study case, Hilbert coding-based approach (HCA), was described in details over Mapreduce.

The performance of the HCA for partitioning big spatial data was test with five different real datasets. And the approach was also compared against the data partition algorithms based on random sampling using SpatialHadoop. Rather than just sampling to make a data partitioning schedule in most researches, we took more information about the whole spatial dataset into consideration with spatial coding. Based on the Hadoop cluster with unfixed nodes, we test different real datasets and job tasks using three measurement standards, namely, the spatial index quality, data skew, and query performance. Compared with the random sampling based method, our approach based on spatial coding technique can improve the query performance of big spatial data, as well as the data balance in HDFS.

Acknowledgment

The research was funded by ministry of land and resources industry public welfare projects (No: 201511010-06).

References

Abel, D.J., Mark, D.M., 1990. A comparative analysis of some two-dimensional orderings. Int. J. Geogr. Inf. Syst. 4, 21–31.
Abel, D.J., Smith, J.L., 1983. A data structure and algorithm based on a linear key for a rectangle retrieval problem. Comput. Vision. Graph. Image Process. 24, 1–13.
Agrawal, S., Narasayya, V., Yang, B., 2004. Integrating vertical and horizontal partitioning into automated physical database design, In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM, Paris, France, pp. 359–370.
Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., Saltz, J., 2013. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. In: Proceedings of the VLDB Endowment 6, pp.1009–1020.
Aly, A.M., Mahmood, A.R., Hassan, M.S., Aref, W.G., Ouzzani, M., Elmeleegy, H., Qadah, T., 2015. AQWA: adaptive query workload aware partitioning of big spatial data. In: Proceedings of the VLDB Endowment 8, pp. 2062–2073.
Aly, A.M., Elmeleegy, H., Qi, Y., Aref, W., 2016. Kangaroo: Workload-Aware Processing of Range Data and Range Queries in Hadoop, In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, San Francisco, California, USA, pp. 397–406.
Avery, C., 2011. Giraph: Large-scale graph processing infrastructure on Hadoop. In: Proceedings of the Hadoop Summit. Santa Clara, 11.
Bajerski, P., Kozielski, S., 2009. Computational Model for Efficient Processing of Geofield Queries, In: Proceedings of the International Conference on Man-Machine Interactions, Kocierz, Poland, pp. 573–583.
Bajerski, P., 2008. Optimization of geofield queries, In: Proceedings of the International Conference on Information Technology, pp. 1–4.
Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B., 1990. The R*-tree: an efficient and robust access method for points and rectangles, In: Proceedings of the 1990 ACM SIGMOD international conference on Management of data. ACM, Atlantic City, New Jersey, USA, pp. 322–331.
Cary, A., Sun, Z.G., Hristidis, V., Rishe, N., 2009. Experiences on processing spatial data with MapReduce. In: Proceedings of the Scientific and Statistical Database Management, 5566, pp. 302–319.
Eldawy, A., Mokbel, M.F., 2013. A demonstration of spatialhadoop: An efficient MapReduce framework for spatial data. In: Proceedings of the VLDB Endowment 6, pp. 1230–1233.
Eldawy, A., Mokbel, M.F., 2015. SpatialHadoop: A MapReduce framework for spatial data, In: Proceedings of the 31st IEEE International Conference on Data Engineering. IEEE Computer Society, Seoul, Korea, Republic of, pp. 1352–1363.
Eldawy, A., Alarabi, L., Mokbel, M.F., 2015. Spatial partitioning techniques in SpatialHadoop. In: Proceedings of the VLDB Endowment 8, pp. 1602–1605.
Eltabakh, M.Y., Özcan, F., Sismanis, Y., Haas, P.J., Pirahesh, H., Vondrak, J., 2013. Eagle-eyed elephant: split-oriented indexing in Hadoop, In: Proceedings of the 16th International Conference on Extending Database Technology. ACM, Genoa, Italy, pp. 89–100.
Gaggero, M., Leo, S., Manca, S., Santoni, F., Schiaratura, O., Zanetti, G., CRS, E., Ricerche, S., 2008. Parallelizing bioinformatics applications with MapReduce. Cloud Comput. Its Appl., 22–23.
Hadjieleftheriou, M., Hoel, E., Tsotras, V.J., 2005. SaIL: a spatial index library for efficient application integration. GeoInformatica 9, 367–389.
Hawick, K.A., Coddington, P.D., James, H.A., 2003. Distributed frameworks and parallel algorithms for processing large-scale geographic data. Parallel Comput. 29, 1297–1333.
Hilbert, D.W., Swift, D.M., Detling, J.K., Dyer, M.I., 1981. Relative growth rates and the grazing optimization hypothesis. Oecologia 51, 14–18.
Hungershöfer, J., Wierum, J.-M., 2002. On the quality of partitions based on space-filling curves, Computational ScienceICCS 2002. Springer, pp. 36–45.
Kitchin, R., 2014. Big Data, new epistemologies and paradigm shifts. Big Data Soc. 1, 1–12.
Liu, L., 2013. Computing infrastructure for big data processing. Front. Comput. Sci. 7, 165–170.
Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M., 2012. Distributed GraphLab: a framework for machine learning and data mining in the cloud. In: Proceedings of the VLDB Endowment 5, pp. 716–727.
Ma, L., Zhang, X., 2007. A computing method for spatial accessibility based on grid partition, Geoinformatics 2007: Geospatial Information Science.SPIE, Nanjing, China pp. 675317–675326.
Meng, L., Huang, C., Zhao, C., Lin, Z., 2007. An improved Hilbert curve for parallel spatial data partitioning. Geo-Spat. Inf. Sci. 10, 282–286.
Miller, H.J., Goodchild, M.F., 2014. Data-driven geography. GeoJournal 80, 449–461.
Minasny, B., McBratney, A.B., Walvoort, D.J.J., 2007. The variance quadtree algorithm: Use for spatial sampling design. Comput. Geosci. 33, 383–392.
Scheuermann, P., Weikum, G., Zabback, P., 1998. Data partitioning and load balancing in parallel disk systems. VLDB J. 7, 48–66.
van Oosterom, P., Vijlbrief, T., 1996. The spatial location code, In: Proceedings of the 7th international symposium on spatial data handling, Delft, The Netherlands.
Vo, H., Aji, A., Wang, F., 2014. SATO: a spatial data partitioning framework for scalable query processing, In: Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, Dallas, Texas, pp. 545–548.
Wei, H., Du, Y., Liang, F., Zhou, C., Liu, Z., Yi, J., Xu, K., Wu, D., 2015. A k-d tree-based algorithm to parallelize Kriging interpolation of big spatial data. Giscience Remote Sens. 52, 40–57.
Ye, J., Chen, B., Chen, J., Fang, Y., Wu, L., 2011. A spatial data partition algorithm based on statistical cluster, Geoinformatics, 2011 In: Proceedings of the 19th International Conference on, pp. 1–6.
Zhao, L., Chen, L., Ranjan, R., Choo, K.-K.R., He, J., 2016. Geographical information system parallelization for spatial big data processing: a review. Clust. Comput. 19, 139–152.