# Spatial Data Dynamic Balancing Distribution Method Based on the Minimum Spatial Proximity for Parallel Spatial Database

Yan Zhou
College of Automation University of Electric Science and Technology of China, Chengdu, P.R. China
zhouyan_gis@yahoo.com.cn

Qing Zhu and Yeting Zhang
State Key Laboratory of Information Engineering in Surveying Mapping and Remote Sensing,
Wuhan University, Wuhan, P.R. China
zhuq66@263.net, zhangyeting@263.net

***Abstract*—Spatial data balancing distribution can evidently improve the performance of parallel spatial database in shared nothing parallel architecture. Considering spatial locality and unstructured variable length characteristics of spatial data, this paper proposes a dynamic spatial data balancing distribution method under shared nothing parallel database environment. By using Hilbert ordering code to keep spatial locality relationship between spatial objects, the presented method can fulfill spatial data static balancing distribution status, which depends on hierarchically decomposing Hilbert space-filling curve code to allocate approximately even spatial data volume to parallel nodes in distributed network. Then spatial proximity index is introduced to resolve spatial data unbalancing problem caused by spatial database dynamic updating. Through moving spatial data fragments to goal node which is the minimum spatial proximity with data moving out node, the spatial database redistributing strategy can attain dynamic data balance among all network parallel nodes. Our experimental results show that the proposed method can effectively improve parallel performance deterioration resulted from spatial data unbalancing and achieve spatial data dynamic balancing distribution in parallel spatial database.**

***Index Terms*—Spatial data distribution; data balancing; parallel spatial database; dynamic data moving; data skew**

## I. INTRODUCTION

Parallel spatial database has been becoming the inevitable trend of high performance spatial database development. The most research of parallel spatial database has focused on shared nothing architecture because of its availability, scalability and high cost performance ratio. Shared nothing parallel architecture means each parallel database node holds exclusive CPU, memory and secondary storage, which can provide parallel dataflow to exploit the I/O bandwidth of multiple disks by reading and writing them in parallel [1]. However, spatial data unbalancing distribution can severely degrade the performance of parallel spatial database in shared noting parallel environment. So the research of spatial data balancing distribution method has caused more and more interests, it means to allocate spatial data to different network nodes uniformly to obtain higher speedup performance of parallel spatial database. Data balancing distribution is one of the most important factors to improve the performance of parallel spatial database under shared nothing parallel architecture [2].

Up to now, a number of data distribution methods have been proposed. Reference [3] summarized several typical data distribution methods (Grid, CMD and HCAM) and discussed how to use them to deal with spatial data allocation. Oracle database also provides two spatial data distributing strategies: one based on X or Y-coordinates value, other based on X and Y-coordinates value [4]. Furthermore, reference [5] proposed a spatial data distribution method called HCSDP which designed specially for spatial data types. All these methods can achieve well data balancing distribution through static data allocating ways. However, static data balance is easy to be broken when facing to dynamic update of spatial database. Many studies have already addressed dynamic balancing distribution issues of parallel database. Reference [6] designed a dynamic bucket spreading strategy in super database system. Reference [7] dealt with data unbalancing issue using partition tuning ways. Reference [8] considered the causes and characteristics of data skew and described a dynamic data balancing method in very large shared nothing hypercube database systems. Reference [9] researched a full dynamic partitioning approach faced to data skew issues that can effectively distribute load among processing nodes without priori knowledge of data distribution, and [10] provided NBRADJUST and REORDER data balancing methods. But these data balancing distribution strategies almost always are under the assumption of uniform data distribution. Unfortunately, spatial data have unstructured

Corresponding author: Yan Zhou.


doi:10.4304/jsw.6.7.1337-1344

variable length and non-uniform distribution characteristics. This means existing dynamic data balancing methods are hardly to be used for resolving spatial data balancing distribution issues. These motivate our research into a proper spatial data dynamic balancing distribution method for shared nothing parallel spatial database.

Our research based on shared nothing parallel database to propose a dynamic spatial data balancing distribution method which includes two main strategies: static data partitioning strategy and dynamic data moving strategy. Section II introduces the static data partitioning strategy. Static data partitioning is aim to achieve initial static data balancing distribution among parallel nodes. Section III presents dynamic data moving strategy. Dynamic data moving strategy is responsible for rebalancing data distribution when spatial database dynamically updating. Section IV gives the experiments and discusses the results. Section V is the conclusion.

## II. STATIC DATA PARTITIONING STRATEGY

Spatial data is unstructured variable length data, and has non-uniform distribution and spatial locality relationship characteristics, so spatial data balancing is more difficult than general data balancing, which need not only satisfy data volume balancing distribution on each parallel node, but also keep spatial locality of spatial objects after data distributing.

### A. Basic idea of static data partitioning strategy

Our static data partitioning strategy is to use initial order Hilbert space-filling curve to impose a linear ordering on multidimensional spatial objects, and then partition spatial objects according to this ordering to preserve spatial locality of spatial objects. To partition whole space into coarse grid cells, and take the Hilbert code as spatial object code when the center point of spatial object find in the interior of corresponding grid cell. Many spatial objects could share the same code. To sum up data volume of spatial objects in each grid cell according to the order of Hilbert code until the cumulative data volume is more than average data volume $V_{ave}$, obviously, it satisfies (1).

$$V_{ave} = V_{total} / P \tag{1}$$

Here, $V_{total}$ denotes the total data volume of spatial objects. $P$ denotes the number of parallel database nodes.

Subsequently, to decompose the last cumulative grid cell into four sub-grid cell and recalculate the Hilbert code of sub-grid. To add spatial object data volume of sub-grids to the cumulative data volume orderly, if need, hierarchical decomposition process of grid cell should be carried out until the cumulative data volume approximately equal to $V_{ave}$. Due to the cumulative data volume could infinitely closed to the average data volume by the hierarchical decomposition of grid cell, there is need to set a final order of Hilbert curve to end the course of hierarchical decomposition of grid cells when the current order of Hilbert curve equal to or more than the given finial order. Setting final order of hierarchical decomposition could avoid excessively intensive space partitioning and improve spatial data partitioning performance.

### B. Two key problems

According to the basic idea mentioned above, static data partition strategy need resolve two key problems: one is how to estimate the initial and final order of Hilbert curve; other is how to encode the decomposed grid cells according to structure features of Hilbert curve.

(1) Hilbert initial and final partitioning order number

Let $n$ denotes the total number of spatial objects. It's easy to understand the course of Hilbert curve hierarchical decomposition should be terminated when each Hilbert grid cell includes only one spatial object at most. That is to say, the number of grid cells should be more than the number of spatial objects. According to the structure of Hilbert curve, $M$ order Hilbert curve has $2^M \times 2^M$ grid cells. $M$ and $n$ should accord with the condition $n < 2^{2M}$, i.e., $M > \frac{1}{2}\log_2 n$. So here, the finial order number of Hilbert curve partitioning is set as (2).

$$M = \left\lceil \frac{1}{2}\log_2 n \right\rceil + 1 \tag{2}$$

Theoretically speaking, the initial order number $m_0$ could be arbitrary positive integer which just need the condition $m_0 < M$, however, if $m_0$ is too small, it will result in overfull hierarchical decomposition, whereas, if $m_0$ is too large, it could be unnecessary. So according to our practical experience, the value of initial order is suggested as (3).

$$m_0 = \lceil M / 2 \rceil \tag{3}$$

(2) Hilbert hierarchical decomposition coding

Because of the fractal feature of Hilbert curve, the *kth* order Hilbert curve always could be generated through finite time's decomposition of the $(k-i)th$ order Hilbert curve. There have been many coding methods of Hilbert curve to be achieved [11-15]. Reference [16] proposed a classical Hilbert coding algorithm which could calculate Hilbert code according to the row and column code of grid cell.

Our hierarchical decomposition method of Hilbert curve is to divide current grid cell into four equal-sized sub-squares cells, which means to halve the row and column of grid cell by horizontal and vertical two directions. If we represent the row and column of grid cell to be $I$ and $J$, then after once hierarchical decomposition, the new row number should be $2I$ and $2I+1$ from the left to right, and the new column number should be $2J$ and $2J+1$ from the bottom to top. At the

same time, the current order of Hilbert curve changes into $m = m+1$. Then based on the classical Hilbert coding algorithm, it's easy to encode the decomposed Hilbert grid cells. Fig.1 illustrates the implementation of hierarchical decomposition from one order Hilbert curve into two order Hilbert curve about spatial object A.

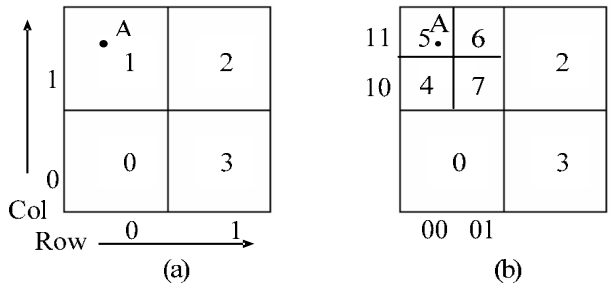



Figure 1. Hilbert hierarchical decomposition coding

### C. Static data partitioning algrithom

TABLE I. SYMBOL DEFINITIONS

| Symbol | Definition |
|---|---|
| $P$ | Number of parallel nodes. |
| $n$ | Number of spatial objects |
| $m_0$ | Initial partitioning order number |
| $m$ | Current order number of Hilbert curve |
| $k$ | Code of $m$ order Hilbert curve, $k \in [0,...,2^{2m}-1]$ |
| $M$ | Final partitioning order number |
| $u_k$ | Number of spatial objects in the *k-th* Hilbert code grid. |
| $v_i$ | Size of the *i-th* spatial object, $i \in [0,...,n]$ |
| $v_a$ | Size of non-spatial attributes of spatial object |
| $B_j$ | Size of spatial objects in the *j-th* partitions, $j \in [1,...,N]$ . |
| $V_k$ | Size of spatial objects in the *k-th* Hilbert code grid |
| $V_{ave}$ | Average data volume of each partition |

To define some symbols used in Table I. Detailed static data partitioning algorithm is described as follow.

- Construct initial Hilbert curve to partition the whole space into grid cells: Initialize $m_0$ and $M$, and divide the whole space range into $I \times I$ sub-grid cells ( $I = 2^{m_0}$ ).
- Code and sort each spatial object according to the Hilbert code of the center point of spatial object.
- Calculate the $V_k$ size of each sub-grid and $V_{ave}$ by (4) and (5).

$$V_k = u_k \times v_a + \sum_{i=1}^{u_k} v_i \qquad (4)$$

$$V_{ave} = V_{total} / P = \sum_{i=0}^{2^{2m}-1} V_k / P \qquad (5)$$

- Sum up data volume of each grid cell according to the order of Hilbert code until the cumulative data volume is more than $V_{ave}$ : Setting $B_j = 0$ , $m = m_0$ , from the beginning of $k = 0$, $j = 1$:

  If $(B_j + V_k) < \overline{V}$ ,
  then $B_j = B_j + V_k$, $k = k+1$;
  Else if $(B_j + V_k) = \overline{V}$ ,
  then $B_j = B_j + V_k$, $j = j+1$, $k = k+1$;
  Else if $(B_j + V_k) > \overline{V}$ and $m < M$ ,
  then $m = m+1$ , and to decompose the current sub-grid into the $m-th$ order grid and calculate the corresponding $V_k$ value of each decomposed grid cell, and then recursively carry out this step until $m = M$ , set $B_j = B_j + V_k$, $j = j+1$, $k = k+1$; when $j > N$ , to stop the course.

- Map partitions $B_j$ to the *j-th* physical storage node.

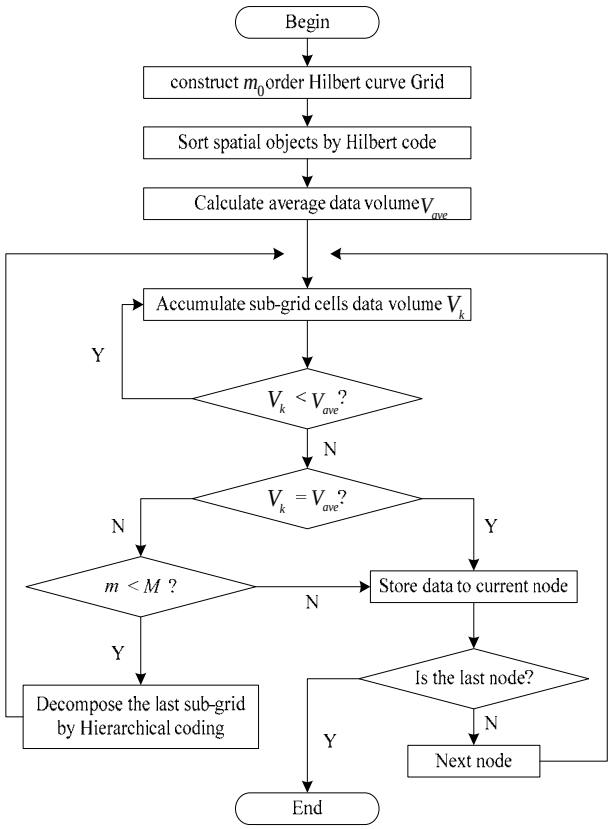



Figure 2. Static data partitioning algorithm.

Thus, spatial objects of cumulative grid cells form the first spatial partition. The rest of partitions could be partitioned by the same iterative processes. By far, spatial data static balance is achieved through hierarchal decomposition of lower order Hilbert grid cell. Fig.2 shows the flowchart of static data partitioning algorithm. The more detail could find in the author's related research paper [17].

## III. DYNAMIC DATA MOVING STRATEGY

### A. Basic idea of dynamic data moving strategy

Static data balance is easy to be broken by inserting or deleting of spatial objects in spatial database. So a dynamic data moving strategy is proposed in here to rebalance spatial data distribution when database updating. In order to avoid damage of spatial locality relationship, our dynamic data moving strategy is to transfer spatial objects based on proximity of spatial datasets in parallel nodes. Proximity is an index measuring spatial locality of spatial objects. Different proximity measure is suitable to different application. Reference [18] defined a measure of similarity of two rectangles $R$ and $S$ is the proportion of queries that retrieve both rectangles as (6).

$$proximity(R,S)=\frac{|q|}{|Q|} \tag{6}$$

The concrete computation of equation could see corresponding reference. Here, $|q|$ is the number of queries retrieving both $R$ and $S$; $|Q|$ is total number of queries. Considering spatial range query is the most common application in spatial database, if using the minimum boundary rectangular of spatial dataset to denote spatial dataset self, then the proximity of two spatial datasets become easy to measure.

Let $P_i$ denotes the *i-th* parallel node with *k* number spatial data subsets $R=\{R_1,R_2,...,R_k\}$, $R_0$ denotes a moving subset, the proximity of $R_0$ and node $P_i$ could be defined as (7).

$$proximity(R_0,P_i)=\max_{R_i\in R} proximity(R_0,R_i) \tag{7}$$

In order to keep the parallel performance of parallel system, the similarity of two spatial datasets should be as low as possible. That is to say, the spatial dataset should move to the candidate node that has the minimum spatial proximity with the current spatial dataset. According to the minimum proximity principle, if select a subset $R_k$ from candidate dataset $R=\{R_1,R_2,...,R_m\}$ to move to the appointed node $P_0$, then $R_k$ should satisfy condition (8).

$$proximity(R_k,P_0)=\min_{R_i\in R} proximity(R_i,P_0) \tag{8}$$

Because data moving between parallel nodes could consume large bandwidth, so generally strategy is to set an experiential threshold value to control the happen frequency of data moving process. The threshold value could be designated according to user requirements and working practices.

### B. Dynamic data moving algorithm

TABLE II. SYMBOL DEFINITIONS

| Symbol | Definition |
|---|---|
| $P_i$ | The *i*-th node. |
| $Pskew_i$ | Data skew of node $P_i$. |
| *Skew* | Data skew of the whole system. |
| $\lambda$ | Threshold |
| $V_i$ | Data volume of node $P_i$. (unit: Byte) |
| $V_{ave}$ | Average data volume of partitions. (unit: Byte) |
| $V_i^{'}$ | The adjustable data volume of node Pi. (unit: Byte) |
| $R_i$ | The *i-th* subset waiting to handling. |

To define some symbols used in Table II. Here, the means of $V_{ave}$ is the same as in (1), and $Pskew_i$ is defined as (9).

$$Pskew_i=\frac{V_i-V_{ave}}{V_{ave}} \tag{9}$$

For a parallel system which has *P* nodes ( $P\in[1,2,...N]$ ), to define the data skew of whole system as (10).

$$Skew=\max_{i=1,2,...P}\{|Pskew_i|\} \tag{10}$$

The proposed spatial data dynamic balancing distribution algorithm is described as follow. Fig.3 shows the flowchart of dynamic data moving algorithm.

- Compute $Pskew_i$ of each nodes and *Skew*, if $Pskew_i<0$, the *i-th* node is Data-In type node, if $Pskew_i>0$, the *i-th* node is Data-Out type node.
- When $Skew\ge\lambda$, to compute $V_i^{'}=|V_i-V_{ave}|$, and Descending $Pskew_i$ and subsets of each node to find a node $P_k$ which has the maximum data skew.
- If $V_k^{'}>0$ ($V_k^{'}=|V_k-V_{ave}|$), to find Data-In nodes ( $Pskew_i<0$ ), and iteratively search subset $R_k$ which is nearest to $\min(V_k^{'},V_i^{'})$ and less than $V_k^{'}$, and then to put $(R_k,P_i)$ into candidate data moving subsets. If candidate subsets is null, to split the maximum subset $R_{\max}$ of node $P_k$ using static data partitioning algorithm described in section II to form proper size candidate subsets.
- If $V_k^{'}<0$, to find Data-Out nodes ( $Pskew_i>0$ ), and iteratively search subset $R_i$ which is nearest to $V_i^{'}$ and less than $R_i$, and then to put $(R_i,P_k)$ into candidate data moving subsets. If

candidate subsets is null, to split the maximum subset $R_{max}$ of Data-Out nodes using static data partitioning algorithm described to form proper size candidate subsets.

- Compute the proximity of each candidate subset and node $P_k$, then to find out the best candidate subset based on the minimum proximity principle shown in equation (8) to move datasets.
- Do the whole steps until satisfy the given threshold condition.

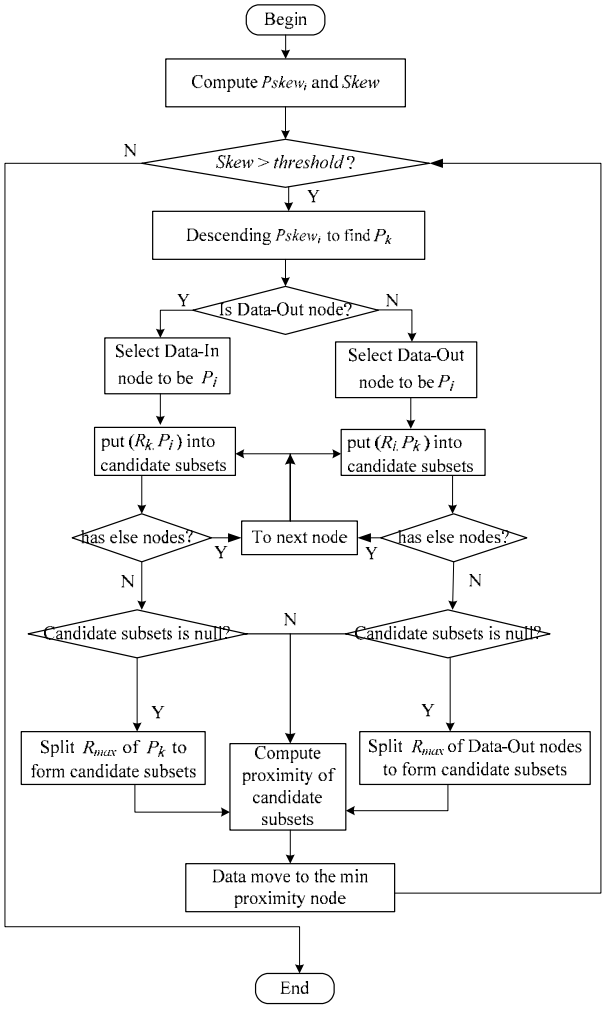



Figure 3. Dynamic data moving algorithm.

## IV. EXPERIMENTS AND RESULTS

Our experiment builds a shared-nothing parallel network environment with 6 general computers, one of these simulates client computer, and the rest simulate parallelism handling nodes. Each computer has the same hardware: CPU 1.8GHz, memory 256M, and connect with 100Mbps Ethernet. The size of experimental dataset is 25,490,352 bytes, include 82,026 spatial objects. The initial distribution of experimental dataset is shown as Table III. Obviously, spatial objects of experimental database are uneven distribution between 5 parallel nodes.

TABLE III. EXPERIMENTAL DATABASET INITIAL DISTRIBUTION (SKEW=0.48330)

| Node No. | Data Volume Distribution （Byte） |
|---|---|
| $P_1$ | 3 581 640 |
| $P_2$ | 3 184 512 |
| $P_3$ | 4 377 528 |
| $P_4$ | 6 784 704 |
| $P_5$ | 7 561 968 |

According to the above mentioned definition of data skew, it's not difficulty to get *Skew* is equal to 0.48330 before data balancing distribution. When setting threshold is equal to 0.1, the processes of data movement and results of dynamic data distributing using the proposed method is shown in Table III. From Table IV, it's easy to see which nodes happened data move in or move out, even the size of data movement when dynamic data moving strategy is executed in every time.

TABLE IV. DYNAMIC DATA MOVING PROCESSES ( $\lambda$ =0.1)

| Executed Times | Data Out | Data In | Data Movement （Byte） | Skew |
|---|---|---|---|---|
| 1 | P5 | P1 | 398 280 | 0.40518 |
| 2 | P5 | P3 | 398 256 | 0.37535 |
| 3 | P4 | P2 | 416 640 | 0.32706 |
| 4 | P5 | P1 | 398 256 | 0.29362 |
| 5 | P4 | P2 | 398 280 | 0.24894 |
| 6 | P5 | P1 | 398 256 | 0.21550 |
| 7 | P4 | P2 | 398 232 | 0.17082 |
| 8 | P5 | P2 | 398 232 | 0.09287 |

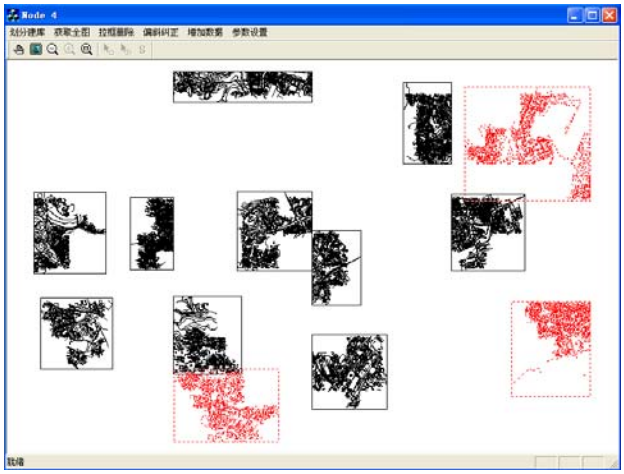


Figure 4. Spatial datasets moving out from Note $P_4$ ( $\lambda$ =0.1).

Fig.4 and Fig.5 illustrate the processes of data dynamic moving algorithm, taking node $P_2$ and $P_4$ for example to show the data dynamic moving in and moving out. Black color denotes spatial datasets stored in local node, green color means datasets moved into current node from other nodes, and red color shows datasets moved out from current node. Obviously, when threshold $\lambda$ is 0.1, $P_2$ is a Data-In node, and $P_4$ is a Data-Out node. Meanwhile, no

datasets of any nodes split. If setting threshold $\lambda$ is 0.05, there are two datasets on node $P_5$ to split into two sub-datasets respectively, as shown in Fig.6, orange color denotes the sub-datasets moved out from current node $P_5$ after splitting, and blue color is the sub-datasets which still stored in local node $P_5$ after splitting.

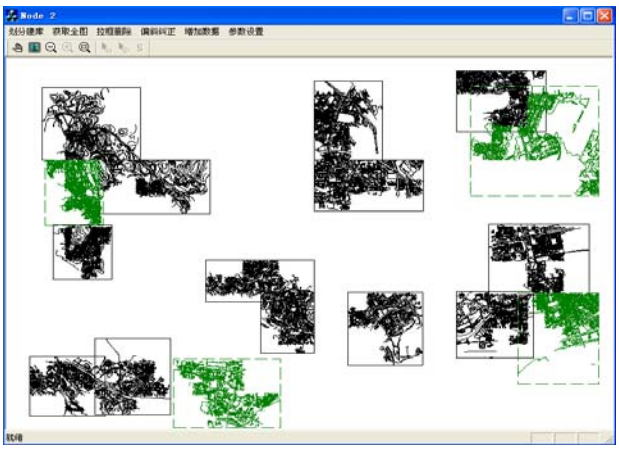


Figure 5. Spatial datasets moving into Note $P_2$ ( $\lambda$ =0.1).

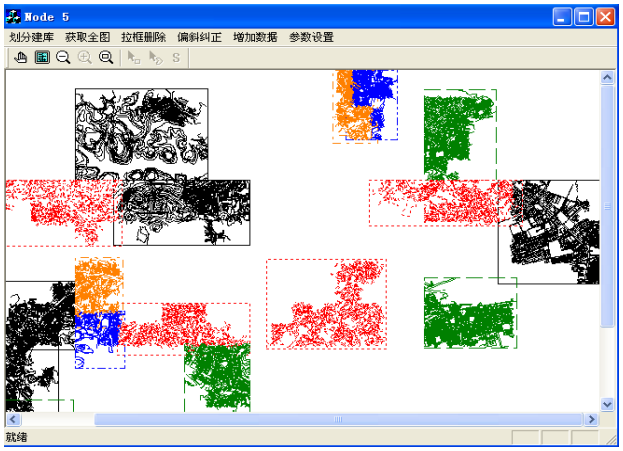



Figure 6. Spatial datasets split into sub-datasets in Note $P_5$ ( $\lambda$ =0.05).

Table V is the results of spatial data redistribution in each node when less than thresholds 0.1, that show each node get more balance distribution. When threshold is 0.1, *Skew* attained 0.09287 after 8 times' dynamic data moving processing, which is less than threshold 0.1. Meanwhile, the distribution of spatial data on parallel nodes from $P_1$ to $P_5$ becomes more even than before. The improvement of data skew before and after using dynamic data moving strategy is shown in Table V.

TABLE V. SPATIAL DATA DYNAMIC BALANCING RESULTS ( $\lambda$ =0.1)

| Node No. | Data Volume Distribution （Byte） | |
|---|---|---|
| | Before (Skew=0.48330) | After (Skew=0.09287) |
| $P_1$ | 3 581 640 | 4 776 432 |
| $P_2$ | 3 184 512 | 4 795 896 |
| $P_3$ | 4 377 528 | 4 775 784 |
| $P_4$ | 6 784 704 | 5 571 552 |
| $P_5$ | 7 561 968 | 5 570 688 |

To find out the relationship between the size of data movement and threshold setting, experiments vary in value of threshold from 0.4 to 0.01 to test the effect of dynamic balancing distribution method, the results of Table VI show that the lower threshold is, the more size of data movement is.

TABLE VI. THE RELATIONSHIP OF DATA MOVEMENT AND THRESHOLD

| Skew Threshold | Executed Times | Data Movement （Byte） | Skew |
|---|---|---|---|
| 0.4 | 2 | 796 536 | 0.37535 |
| 0.3 | 4 | 1 611 432 | 0.29362 |
| 0.2 | 7 | 2 806 200 | 0.17082 |
| 0.1 | 8 | 3 204 423 | 0.09287 |
| 0.05 | 12 | 4 063 455 | 0.02725 |
| 0.02 | 14 | 4 167 519 | 0.01476 |
| 0.01 | 16 | 4 242 687 | 0.00683 |

In order to prove the improvement of parallel operation performance after spatial data dynamic balancing distribution, we test the operation performance of spatial range query before and after data skew handling vary in *Skew* (0.48330 and 0.95516). Without loss of generality, the whole index region is normalized unit square. The query rectangles are squares with size Q, the normalized query range Q is change from 0.2~0.6. Their centers are uniformly distributed in the unit square. For every time experiment, 100 randomly generated queries were asked and the results were averaged as response time of the spatial query. The results of Table VII show that the response time of spatial query when *Skew* is 0.48330 is always better than those of *Skew* 0.95516, which proves the larger data skew is, the lower performance of parallel spatial database is, and vice versa. So, spatial data dynamic balancing distribution method would help to improve spatial query performance of parallel spatial database.

TABLE VII. EFFECT OF DATA SKEW FOR SPATIAL QUERY PERFORMANCE ( $\lambda$ =0.1)

| Query Range | Response Time of Spatial Query (Unit : ms) | | | |
|---|---|---|---|---|
| | Skew=0.48330 | | Skew=0.95516 | |
| | Before | After | Before | After |
| 0.2 | 25 | 24 | 27 | 25 |
| 0.3 | 50 | 47 | 55 | 49 |
| 0.4 | 85 | 83 | 94 | 84 |
| 0.5 | 133 | 130 | 145 | 131 |
| 0.6 | 190 | 183 | 198 | 188 |

From the results of Table IV to Table VII, all these experiments prove that the proposed spatial data dynamic balancing distribution method could satisfy spatial data dynamic balancing distributing requirements effectively, and improve spatial query performance obviously. Our proposed method is an effective spatial data dynamic balancing distribution method.

## V. CONCLUSION

Spatial data unbalancing distribution could degrade the performance of parallel spatial database under shared nothing parallel environment. Considering spatial locality and unstructured variable length characteristics of spatial data, this paper proposed a dynamic spatial data balancing distribution method based on the minimum spatial proximity which consists of static data partitioning strategy and dynamic data moving strategy. Our experimental results show that the proposed method could effectively achieve spatial data dynamic balancing distribution.

## ACKNOWLEDGMENT

This research supported by the National High Technology Research and Development Program of China (No. 2008AA121602), National Natural Science Foundation of China (No. 40871212 and 40701144) and Open Research Fund of State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (No. 08I02).

## REFERENCES

[1] Valduriez, P., "parallel database systems: open problems and new issues", *Distributed and Parallel Databases*, vol. 1(2), 1993, pp.12-19.

[2] Rahm, E. and R. Marek., "Analysis of Dynamic Load Balancing Strategies fro Parallel shared Nothing Database Systems", *in Proceedings of the 19th VLDB Conference*, 1993, pp.182-193.

[3] Shekhar S and Chawl S., *Spatial Databases A Tour*, China Machine Press, Beijing, 2004.

[4] Abugov D, *Oracle Spatial Partitioning: Best Practices (an Oracle White Paper)*, Oracle Inc., 2004.

[5] Zhao C Y, Meng L K, Lin Z Y., "Spatial Data Partitioning Towards Parallel Spatial Database System," *Geomatics and Information Science of Wuhan Univeristy*, vol. 31(11), pp. 391-394. (in Chinese), 2006.

[6] Kitsuregawa, M. and Y. Ogawa., "Bucket spreading parallel hash: a new, robust, parallel hash join method for data skew in the super database computer ", *in Proceedings of the 16th VLDB Conference*, Brisbane, Australia, 1990, pp. 210-221.

[7] Hua, K. A. and C. Lee., "Handling data skew in multiprocessor database computers using partition tuning", *in Proceedings of the 17th Very Large Data Bases*, Barcelona, Spain, 1991, pp. 525-535.

[8] Hua, K. A. and J. X. W. Su., "Dynamic load balancing in very large shared-nothing hypercube database computers", *IEEE Transactions on Computer*, vol. 42 (12), 1993, pp.1425-1439.

[9] Lu, H. and J. X. Yu, et al., "Fully Dynamic Partitioning: Handling Data Skew in Parallel Data Cube Computation." *Distributed and Parallel Databases*, vol.13 (2), 2003, pp.181-202.

[10] Ganesan, P. and M. Bawa, et al., "Online balancing of range-partitioned data with applications to peer-to-peer systems ", *in Proceedings of the 30th Very Large Data Bases Conference*, Toronto, Canada, 2004, pp.444-455.

[11] Lu F, Zhou C H., "An Algorithm for Hilbert Ordering Code Based on Spatial Hierarchical Decomposition", *Journal of Image and Graphics*, vol. 6(A) (5), 2001, pp.465-469.

[12] Bially T., "Space-filling curves: Their generation and their application to bandwidth reduction", *IEEE Transactions on Information Theory*, vol. 15(6), 1969, pp.658-664.

[13] Liu X, Schrack G., "Encoding and decoding the Hilbert order", *Software-Practice & Experience*, vol.26 (12), 1996, pp.1335-1346.

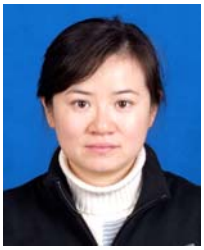
[14] Liu X, Schrack G., "A new ordering strategy applied to spatial data processing", *International Journal of Geographical Information Science*, vol. 12(1), 1998, pp.3-22.

[15] Bartholdi J. J, Goldsman P., "Vertex-labeling algorithms for the Hilbert spacefilling curve", *Software-Practice & Experience*, vol. 31(5), 2001, pp.395-408.

[16] Faloutsos C, Roseman S., "Fractals for Secondary Key Retrieval", I*n Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, Philadelphia, Pennsylvania, United States, 1989, pp.247-252.

[17] Zhou, Y. and Q. Zhu, et al., "A GIS Spatial Data Partitioning Method for Distributed Data Processing", *in MIPPR 2007: Remote Sensing and GIS Data Processing and Applications; and Innovative Multispectral Technology and Applications*, edited by Yongji Wang, Jun Li, Bangjun Lei, Jingyu Yang, Wuhan, China, Proc. of SPIE. 6790, 2007, pp. 679008 1-7.

[18] Kamel, I., and C. Faloutsos, "Parallel R-trees", *in Proceedings of the 1992 ACM SIGMOD international conference on Management of data*, San Diego, California, United States, 1992, pp.195-204.



**Yan Zhou** was born in Shaanxi, China, in October 1976. She received her BS degree in information engineering in 1999, Master degree in geographic information engineering in 2004 and Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, Hubei, China in 2008.

She is a lecturer in the College of Automation University of Electric Science and Technology of China, Chengdu, Sichuan, China.

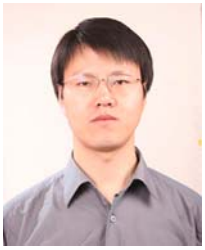
Dr. Zhou's research interests and publications have been focused on distributed GIS, high performance geo-computing and parallel spatial databases technology.

**Qing Zhu** was born in Sichuan, China, in July 1966. He received his BS degree in 1986 and Master degree in 1989 in railway engineering from Southwest Jiaotong University, Chengdu, Sichuan, China. He received his Ph.D. degree in railway engineering from Northern Jiaotong University, Beijing, China in 1995.

He is a Professor in the State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing at Wuhan University, Wuhan, Hubei, China. His main research interests are digital

photogrammetry, 3D GIS and virtual geographic environments.

Prof. Zhu was the Co-Chair of ISPRS Working Group VI/5 “Promotion of the Profession to Students” from 2004 to 2008 and the Co-Chair of ISPRS Working Group V/4 “Image-based and range-based 3D modelling” from 2008 to 2012. He is the Editorial Board member of 《Computers, Environment and Urban Systems》.



**Yeting Zhang** was born in Shanxi, China, in March 1978. He received his M.S. degree in geographic information system from Wuhan University, Wuhan, Hubei, China in 2002, and Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, Hubei, China in 2008.

He is a lecturer in the State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing at Wuhan University, Wuhan, Hubei, China.

Dr. Zhang's main research interests are spatial database technology, 3D GIS and virtual geographic environments.