Declustering Spatial Objects by Clustering for Parallel Disks

Hak-Cheol Kim and Ki-Joune Li

Dept of Computer Science, Pusan National University, Korea
Email: hckim@quantos.cs.pusan.ac.kr, lik@hyowon.pusan.ac.kr

Abstract. In this paper, we propose an efficient declustering algorithm which is adaptable in different data distribution. Previous declustering algorithms have a potential drawback by assuming data distribution is uniform. However, our method shows a good declustering performance for spatial data regardless of data distribution by taking it into consideration. First, we apply a spatial clustering algorithm to find the distribution in the underlying data and then allocate a disk page to each unit of cluster. Second, we analyze the effect of outliers on the performance of declustering algorithm and propose to handle them separately. Experimental results show that these approaches outperform traditional declustering algorithms based on tiling and mapping function such as DM, FX, HCAM and Golden Ratio Sequence.

1 Introduction

Spatial database systems, such as geographic information systems, CAD systems, etc., store and handle a massive amount of data. As a result, frequent disk accesses become a bottleneck of the overall performance of system. We can solve this problem by partitioning data onto multiple parallel disks and accessing them in parallel. This problem is referred to as declustering. For parallel disk accesses, we divide the entire data set into groups by the unit of disk page and assign a disk number so that partitioned groups should be accessed at the same time by a query.

Up to now, several declustering methods have been proposed. Most of them partition a data space into several disjoint tiles and match them with a disk number using mapping function[1, 2, 4, 14, 16]. They focussed only on an efficient mapping function from a partitioned tile to a disk number on the assumption that data is uniformly distributed. Therefore their methods, though give a good performance for uniform data, show a drop in efficiency for skewed data.

To be effective for a skewed data, a declustering algorithm must be reflected of the distribution in the underlying data. We can apply two approaches, parametric and nonparametric methods, to discover distribution. By the parametric method, we assume a parametric model such as normal distribution, gamma distribution, etc. However, we cannot adopt this approach because most of the real data distributions do not agree with these distribution models. Another approach, non-parametric methods, makes no assumption about distribution.

Several methods such as kernal estimation, histogram, wavelet and clustering methods belong to this category. Among these methods, we apply a spatial clustering method called SMTin[6] to detect data distribution.

By applying a spatial clustering method, our method is more flexible with the distribution than tiled partitioning methods and results in a high storage utilization and a low disk access rate. In addition to this contribution, we analyze the effect of outliers on the performance of declustering algorithm and propose a simple and efficient method to control them.

This paper is organized as follows. In section 2, we present related works and their problems and propose a new declustering algorithm in section 3. In section 4, we show the effects of outliers on the performance of declusteirng algorithm and our solutions. We show experimental results in section 5 and conclude this paper in section 6.

2 Background and Motivation

Since the performance of spatial database system is more affected by the I/O cost than CPU cost, a great deal of efforts have been done to partition data onto multiple parallel disks and access those object qualifying query condition simultaneously. Traditionaly this problem is called as declustering, which consists of the following two steps.

- step 1: grouping data set by the unit of disk page size
- step 2: distributing each group of data onto multiple disks

When we distribute objects onto multiple parallel disks, the response time of query q is determined as follows.

Definition 1. *Let M be the number of disks, then the number disk accesses to process query q is*

$$DA_q = \max_{i=1}^{M}(DA_q(i)), \text{ where } DA_q(i) \text{ is the number of i-th disk access}$$

Based on definition 1, Moon and Saltz defined the condition of the strict optimality of declustering algorithm[13].

Definition 2. Strictly Optimal Declustering
Let M be the number of disks, $DA_q(i)$ is the number of i-th disk access for a query q, then declustering method is strictly optimal if

$$\forall q,\ DA_q = \lceil \frac{N_q}{M} \rceil, where \quad N_q = \sum_{i=1}^{M} DA_q(i)$$

Previously proposed declustering methods tile a data space and assign disk number to each tile based on a certain mapping function. Disk Modulo(DM)[1], Field-wise Xor(FX)[2] and Hilbert Curve Allocation Method(HCAM)[4] use this scheme. Among these methods, Hilbert Curve Allocation Method proposed by C. Faloutsos and P. Bhagwat has been known to outperforms others[4]. Moon and Saltz proved that the scalability of DM and FX is limited to some degree. They proved that the scalability of DM is bounded by the query side and the

scalability of FX is at best 25 percents by doubling the numger of disks. For more details, see [13].

Recently, Bhatia and Sinha proposed a new declustering algorithm based on Golden Ratio Sequences[16]. Their analytical model and experimental results show that GRS outperforms not only traditional tile-based declustering methods such as DM, FX, HCAM, but also cyclic allocation method[14], which is a generalization of DM.

Most of these declustering methods are known to be not stritly optimal without some assumptions[1, 2, 4, 8, 14, 16] and try only to find an optimal way in allocating each tile onto parallel disks on the assumption that data is uniformly distributed. However, they ignore the effects of a good grouping method on increasing the storage utilization for step 1. It is obvious that the maximum number of page accesses per disk grows as the total number of disk page occupied by objects increases. As a result, we might access more disk pages without a good grouping method.

In addition to this drawback, they did not consider outliers having an effect on the performance of declustering algorithm. We will explain its effects more closely in section 4.

3 Declustering Skewed Dataset

In section 2, we mentioned that previously proposed declustering algorithms have potential weakness by assuming uniform data distribution. we will show it in detail and propose a new declustering algorithm in this section.

3.1 Skewed Data and Declustering

First, we will describe how the performance of a tiling algorithm can be affected by data distribution. When the number of objects in a certain densed tile exceeds the disk page capacity, all the tiles must be split though they contain small number of objects that can fit into one disk page. This results in a low storage utilization and a poor performance in comparison with uniform data even if a same mapping function is used. We investigate this problem in the rest of this section in detail. Table 1 shows notations and their meanings to be used from now on.

Lemma 1. *When we apply the same tiling scheme, the number of total disk pages occuppied by skewed data is more than that of uniform data.*

Proof. Let n_i be the number of objects in the *i-th* tile occupied by skewed data. Then $\sum_{i=1}^{T_s} n_i = N$. Since the distribution of data is skewed, $\min_{i=1}^{T_s}(n_i) < \max_{i=1}^{T_s}(n_i) = Bf_{max}$ and $\text{avg}_{i=1}^{T_s}(n_i) = N/T_s < Bf_{max}$. For uniform data, the number of objects in a tile is N/T_u and if the storage utilization is maximum, the number of objects in a tile is Bf_{max}. It means that $N/T_u = Bf_{max}$. Therefore $N/T_s < N/T_u$ that is $T_s > T_u$. ■

Notation	Meaning
N	number of spatial objects in 2-D space
T_u, T_s	total number of tiles(disk pages) occupied by uniform data and skew data respectively
Bf_{max}	maximum disk blocking factor
d_u, d_s	density of uniform data and skewed data respectively
a_u, a_s	area of one tile for uniform data and skewed data respectively

Table 1. Notations and their meanings

Lemma 2. *For a given query q, Let $T_u(q)$ and $T_s(q)$ be the numbers of disk page accesses to process query q for uniform data and skewed data respectively. Then,*

$$T_u(q) < T_s(q)$$

Proof. We normalize the data space as $[0,1]^2$. For completely uniform data, the number of disk pages T_u and the area of a tile are given as follows.

$$T_u = \frac{N}{Bf_{max}} \tag{1}$$

$$a_u = \frac{1}{T_u} = \frac{Bf_{max}}{N} \tag{2}$$

Since there are Bf_{max} objects in a tile, we obtain the following equation

$$d_u \cdot a_u = Bf_{max}, \qquad a_u = \frac{Bf_{max}}{d_u} \tag{3}$$

For skewed data, the number of objects contained in a tile is variable. We get the following equation for skewed data.

$$\max(d_s) \cdot a_s = Bf_{max}, \qquad a_s = \frac{Bf_{max}}{\max(d_s)} \tag{4}$$

It is evident that $d_u < \max(d_s)$, since $\max(d_s)$ is the maximum density of skewed data. Therefore we derive the following inequality from equation 3, 4.

$$a_s < a_u \tag{5}$$

For a query q whose area is $\mathrm{A}(q)$, the number of tiles contained by q for the uniform data, $T_u(q)$ and skewed data $T_s(q)$ are given as

$$T_s(q) = \frac{A(q)}{a_s}, \qquad T_u(q) = \frac{A(q)}{a_u} \tag{6}$$

From equation 5 and 6, we know that the number of tiles for the skewed data qualifying the same query condition is larger than that of uniform data, since $a_s < a_u$. ■

From lemma 1 and 2, we come to a conclusion that tiling methods for skewed data cannot satisfy the strict optimality condition of defintion 2.

Theorem 1. *Suppose that $DA_u(q)$ and $DA_s(q)$ are the maximum number of page accesses per disk for uniform and skewed data to process query q respectively. Then,*

$$DA_u(q) < DA_s(q)$$

Proof. If we assume an optimal declustering algorithm, the number of page accesses per disk for uniform data and skewed data to process query q is given as follows.

$$DA_u(q) = \lceil \frac{T_u(q)}{N} \rceil, \qquad DA_s(q) = \lceil \frac{T_s(q)}{N} \rceil$$

Since $T_u(q) < T_s(q)$, $DA_u(q) < DA_s(q)$. ■

This means that the number of tiles(disk pages), in other words storge utilization, is an important factor in declustering method in case of skewed data. In fact, we found an important difference between T_s and T_u depending on data and the degree of skewedness from experiments.

These observations lead to a conclusion that it is very important to partition spatial objects in a way that reduces the number of tiles, in addition to find a good mapping function for allocating each tile to a disk number. We will focus on methods to improve the storage utilization in the next subsection.

3.2 An Efficient Declustering Method for Skewed Data

In this subsection, we propose a new declustering algorithm which is flexible to the distribution of data and results in a small number of pages in comparision with precedent declustering algorithms. The proposed algorithm is composed of the following three steps.

Step 1. Find data distribution
In this paper, we apply a spatial clustering algorithm to find the distribution of data. Up to now, several spatial clustering methods have been introduced[5, 6, 10, 9]. Among them, we apply SMTin as a spatial clustering algorithm. SMTin initially constructs delaunay triangulations for point set and extracts clusters from triangles whose distance is within a predefined threshold value. We can find the distribution in the underlying N objects by the time complexity of $O(N\log N)$. For more details, see [6].

Step 2. Split overflow clusters
After clustering step, there may be clusters whose number of elements exceeds disk page capacity. These clusters must be partitioned into several sub-clusters, each of which can fit into one disk page. we applied an efficient partitioning method called STR proposed by Leutenegger, et al.[7].

Step 3. Distribute clusters onto disk page
After adjusting all of clusters so as to fit into one disk page, we calculate the center of minimum bounding rectangle enclosing cluster, sort them by the hilbert value of their center and then assign a disk number in a round robin fashion.

4 Outliers and Declustering

In data analysis, data with large dissimilarity with others may deteriorate result. Many researhes have been done about these data called outlier in data minining area and various definitions of outlier have been made in the literature[11, 12, 15]. But we give a differenct definition of outlier, which is based on SMTin as follows.

Definition 3. Outliers with Cluster Construction Threshold value
Let CCT_{cs} be a cluster construction threshold value and $C_1, C_2, \ldots, C_n$ be clusters by the CCT_{cs}, then any object O' satisfying the following condition is outlier:

$$\forall O \in C_i, 1 \leq i \leq n, dist(O, O') > CCT_{cs}$$

In fact, we can control the number of outliers by means of CCT_{cs}. We will explain the effects of outliers and our solutions in the following subsections.

4.1 The effect of outliers on the declustering

First, we illustrate the problems of outliers. After finding an initial cluster set, there can be some objects not included in any clusters depending on an initial cluster construction threshold value. We have to increase cluster construction threshold value to include them and the shape of cluster may be degenerated and minumum bounding rectangle enclosing cluster may be extended unnecessarily.

Another problem is that these clusters whose size is extremely smaller than maximum page size result in a low storage utilization. Although they may be regarded as a cluster, it is desirable to treat them as outliers.

4.2 An Efficient Method for Skewed Data with Outliers

In section 4.1, we showed that outliers may degenrate the performance and must be carefully treated. We may apply two approaches to handle them. The simplest apporach is include them in the nearest cluster by force. Another solution is to regard outlier as a cluster whose size is extermely small and assign one disk page for it. However, these solutions result in a low storage utilization and a high disk access rate.

We propose to keep the outliers on an extra main-memory buffer. As outlier buffer size has a limitation, we keep a part of outliers in buffer and include the remained outliers in the nearest cluster by force. Figure 1 shows our proposed declustering algorithm.

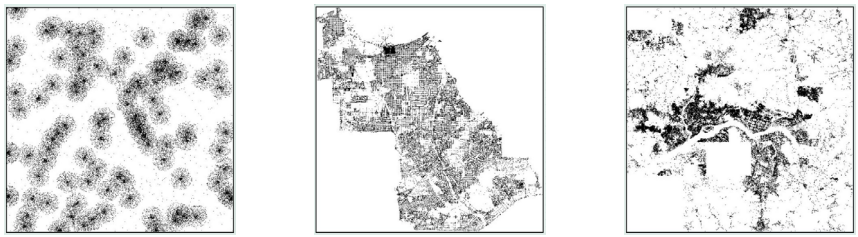
```
Algorithm DC (Declustering Clusters to parallel disks)
Input P: set of points {p1, p2, ..., pn}, M: number of disks available
      Bfout: capacity of outlier buffer, Bf: disk blocking factor
      CCTcs: cluster construction threshold value
Output {Ci, di}  //Ci: i-th cluster, di: disk number assigned to Ci
Begin Algorithm
    C ← {};
    Construct initial clusters by CCTcs;
    While (Card(outlier) > Bfout)
        adjust CCTcs;
        reconstruct clusters;
    End while
    For each cluster Si
        If Card(Si) > Bf
            {Si1, Si2, ..., Sik } ← SplitCluster(Si, Bf);
            C ← C ∪ {Si1, Si2, ..., Sik};
        End If
        Else
            C ← C ∪ {Si};
        End Else
    End For
    For each element Ci in C
        (xi, yi) ← center of cluster Ci;
        di ← H(xi, yi) mod M; // H(x, y) is Hilbert value of (x, y)
    End For
End Algorithm
```

Fig. 1. Description of Declustering by Clustering(DC) algorithm

5 Performance Evaluation

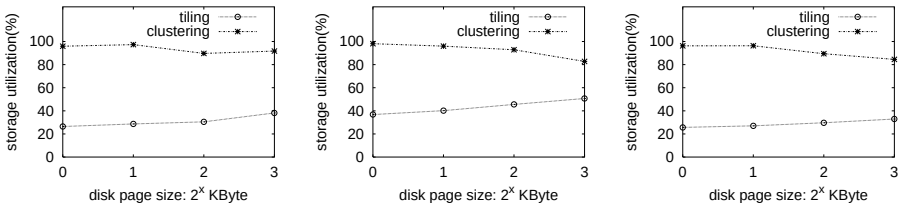
We performed several experiments to compare our method with previously proposed declustering algorithms. To do this, we generated synthetic skewed data to analyize the effect of data distribution on the performance of declustering. We also prepared two real data set extracted from the maps of Long Beach County and Seoul city. Figure 2 shows its distribution. For query set we generated two types of queries, which is uniformly distributed in data space and concentrated on a data area, and whose size is 0.1×0.1(small query) and 0.5×0.5(large query) of data space whose size is $[0, 1]^2$.

Storage Utilization
We explained that the number of total disk pages has an effect on the performance. We found that tiling method occupy more disk pages than our proposed method. In detail, it occupy 1.6∼3.7 times more disk pages than declustering



(a) skew synthetic data (b) LBCounty (c) Seoul

Fig. 2. Test data distribution



(a) skew (b) LBCounty (c) Seoul

Fig. 3. Storage utilization of tiling and clustered partition(outlier buffer is 1 KB)

by clustering algorithm. Figure 3 shows the storage utilization of tiling method and DC. It shows that tiling method is far from being optimal allocation but our method is nearly optimal as far as a storage utilization is concerned.

Scalability

We compared our algorithm only with Golden Ratio Sequence(GRS)[16] since it is known to be the best among tiling methods. Figure 4 shows the result of experiments. We see that our method gives a good performance regardless of data distribution and query size. Especially, when the number of disk is small and query size is large, enhancement of our method over GRS is significant. When query size is 25% of data space and the number of disks is 8, the declustering method by Golden Ratio Sequence accesses about 8 times more disk pages than DC for Seoul data. However, the performance improvement ratio over GRS become lower as the number of available disks increases.

We carried out a simular experiment with a set of queries concentrated on a downtown area rather than uniformly distributed. It is more realistic environment of experiment since queries tend to be located on specific regions. The experiment however shows very similar results with that of uniformly distributed queries.

The effect of outliers

We carried out an experiment to reveal the effects of outliers on the performance of declustering algorithm. If the size of outlier buffer is small, a cluster

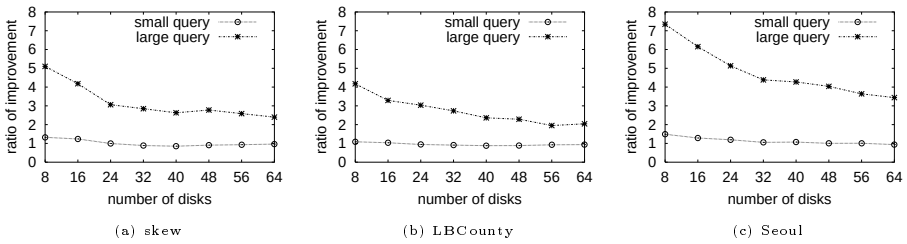


Fig. 4. Performance enhancement ratio of DC over GRS for uniformly distributed queries, where disk page size is 4 KByte and outlier buffer is 1KByte

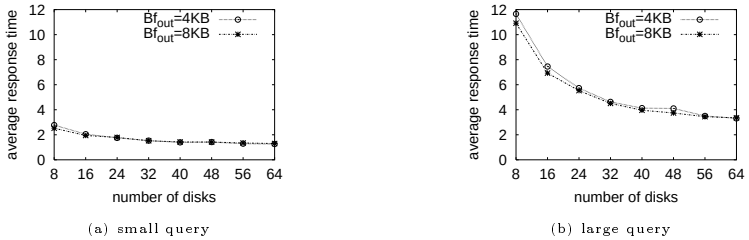


Fig. 5. Average response time by an outlier buffer size: disk page size is 1KByte

construction threshold value becomes large and the minimum bounding rectangle enclosing a cluster must be extended to include outliers. We found that the effect of outlier buffer is affected by a data distribution. It is of no use to increase outlier buffer exceeding a certain value which is related to the data distribution.

Figure 5 shows the effect of a outlier buffer on the performance of the proposed declustering algorithm for Seoul data. We gain more performance enhancement, though its effectiveness is trivial, by increasing the size of outlier buffer.

6 Conclusion

In this paper, we proposed a new declustering method for spatial objects. We investigated the effect of data distribution on the performance of declustering algorithm and showed that previously presented algorithms do not give a good performance for skewed and real data. We reviwed the definition of strict optimality proposed by Moon and Saltz and showed that tiling algorithms cannot be strictly optimal for skewed data.

Before declustering spatial objects onto multiple disks, we apply a spatial clustering algorithm to discover the data distribution. By doing so, our method is more flexible to the data distribution than tiling methods. In addition to this contribution, we showed the effects of outlier on the performance of declustering algorithm and proposed to store them separately.

Experimental results show that our method gives a higher storage utilization and lower disk access rate regardless of data distribution and query distribution.

Currently our works are limited to a static data set. In our future work, we will study on the dynamic clustering algorithm and show the effect of outliers more closely.

Acknowledgments

The author(s) wish(es) to acknowledge the financial support of the Korea Research Foundation made in the Program Year 1997.

References

1. Du, H.C., Sobolewski, J.S.: Disk Allocation for Cartisian Files on Multiple-Disk Systems. Int. J. ACM TODS, Vol. 7, No. 1, (1982) 82-102
2. Fang, M.T., Lee, R.C.T., Chang, C.C.: The Idea of De-Clustering and Its Applications. VLDB (1986) 181-188
3. Faloutsos, C., Metaxas, D.: Disk Allocation methods using error correcting codes. Int. J. IEEE Trans on Computers, Vol. 40, No. 8, (1991) 907-914
4. Faloutsos, C., Bhagwat, P.: Declustering using fractals. Parallel and Distributed Information Systems Conf (1993) 18-25
5. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering methods for very large databases. SIGMOD (1996) 103-114
6. Kang, I.S., Kim, T.W., , Li, K.J.: A spatial data mining method by delaunay triangulation. Proc. ACM-GIS. (1997) 35-39
7. Leutenegger, S. T., Lopez, M.A., Edgington, J.M.: STR: A simple and efficient algorithm for r-tree packing. ICDE. (1997) 497-506
8. Abdel-Ghaffar, K., Abbadi, A. E.: Optimal allocation of two-dimensional data. ICDT (1997) 409-418
9. Sheikhleslami, G., Chatterjee, S., Zhang, A.:Wavecluster: A multi-resolusion clustering approach for very large spatial databases. VLDB (1998) 428-439
10. Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithms for large databases. SIGMOD (1998) 73-84
11. Knorr, E., Ng, R.: Algorithms for Mining Distance-Based Outliers in Large Datasets. VLDB (1998) 392-403
12. Barnett, V., Lewis, T.:Outliers in Statistical Data. Third Edition, John Wiley & Sons Ltd. (1998)
13. Moon, B. K., Saltz, J. H.: Scalability Analysis of Declustering Methods for Multidimensional Range Queries. Int. J. IEEE TKDE, Vol. 10, No. 2, (1998) 310-327
14. Prabhakar, S., Abdel-Ghaffar, K., El Abbadi, A.: Cyclic allocation of two-dimensional data. ICDE (1998) (94-101)
15. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient Algorithms for Mining Outliers from Large Data Sets. SIGMOD (2000) (427-438)
16. Bhatia, R., Sinha, R.K., Chen, C.-M.:Declustering using golden ratio sequences. ICDE (2000) 271-280