ISSN 2395-1621

Metric Access Methods for Distance Cache Mining

Komal Surawase[#1], Sonali Surawase [#2],Anuja Patil[#3]

[1]surawasekomal23@gmail.com

[2]sonalisurawase@gmail.com

[3]patilanuja.48@gmail.com

[#12]Dept. of Computer Engineering, J.S.C.O.E.
Hadapsar, Pune, Maharashtra, India.

[#3]Dept. of Computer Engineering, S.V.I.T
Nashik (MS), Maharashtra, India.

ABSTRACT

This article is related to increase the DBMS performance and resolve all issues and risks. Here we implement the new caching and buffering techniques. These new caching techniques consume the I/O cost utilization. Previous working procedure starts in complex databases. User forward the query, same query result is present in different databases; using similarity operation extracts the results from the distributed databases. All related or relevant results are displayed here. It can have the retrieval performance as very low. Here utilization of I/O cost and CPU cost is high. It can have minor performance under computation cost. Next we have changed the query format like k-nearest neighbor. It can display the results at least 80%. It can have the non-relevant results of information. It is expensive query based data extraction. We are proposing structures related cache distances. Any user forward any kind of query, automatically it can search, run timely and display the results. Run timely in database technology perform the analysis process and provides the results with optimization of I/O cost here. It can work based on distance caches in implementation. It can provide the results in indexing and querying. It can display the effective results. Compare all the previous methods pivot based query provides the effective results

***Keywords*— Metric Access, k-nearest neighbor query, M Tree, D-cache.**

ARTICLE INFO

Article History

Received : 2nd February, 2015

Received in revised form :

5th March, 2015

Accepted :8th March, 2015

Published online :

10th March 2015

I. INTRODUCTION

The Distance cache mining aims to amortize the number of distance computations spent by querying/updating the database, similarly like disk page buffering in traditional DBMSs aims to amortize the I/O cost. The D-cache structure is based on a hash table, thus making efficient to retrieve stored distances for further usage[1],[2],[4],[5]. Additionally, the D-cache maintains the set of previously processed runtime objects (i.e., inserted or query objects), while the most recent of them are used as dynamic pivots.

The D-cache supports three functions useful for metric access methods (MAMs)—the GetDistance (returning the exact distance between two objects, if available), the GetLowerBoundDistance (returning the greatest lower-bound distance between two objects, by means of the dynamic pivots), and the GetUpperBoundDistance (returning the lowest upper bound distance). With these functions, the D-cache may be used to improve the construction of MAMs' index structures and the performance of similarity queries. Our depiction of the D-cache is general, and may be used with any metric access method or even to aid a sequential scan of the database—forming a brand new concept of index-free MAM, the D-file. We have presented replacement policies for the distances stored in the cache as well as algorithms for the computation of the lower- and upper bound distances. We have also described in detail how to enhance some of the existing metric access methods (M-tree, Pivot tables) with the D-cache

II. LITERATURE SURVEY

The Fundamentals of Similarity Search in Metric Spaces During the last two decades, similarity searching in metric spaces has become intensively investigated research area , as documented in several excellent monographs and surveys the next paragraphs , we will shortly summarize basic fields of the content-based similarity searching using metric space model[3]. For more details, see one of the referred publications. [6][7]Content based Image Retrieval The

earliest use of the term content based image retrieval in the literature seems to have been by Kato [1992], to describe his experiments into automatic retrieval of images from a database by color and shape feature. The term has since been widely used to describe the process of retrieving desired images from a large collection on the basis of features (such as color, texture and shape) that can be automatically extracted from the images themselves.

The features used for retrieval can be either primitive or semantic, but the extraction process must be predominantly automatic. Retrieval of images by manually-assigned keywords is definitely not CBIR as the term is generally understood – even if the keywords describe image content [8],[11]. CBIR differs from classical information retrieval in that image databases are essentially unstructured, since digitized images consist purely of arrays of pixel intensities, with no inherent meaning [9]. One of the key issues with any kind of image processing is the need to extract useful information from the raw data (such as recognizing the presence of particular shapes or textures) before any kind of reasoning about the image's contents is possible. Image databases thus differ fundamentally from text databases, where the raw material (words stored as ASCII character strings) has already been logically structured by the author [Santini and Jain, 1997]. There is no equivalent of level 1 retrieval in a text database. CBIR draws many of its methods from the field of image processing and computer vision, and is regarded by some as a subset of that field [10], [2]. It differs from these fields principally through its emphasis on the retrieval of images with desired characteristics from a collection of significant size. Image processing covers a much wider field, including image enhancement, compression, transmission, and interpretation. While there are grey areas (such as object recognition by feature analysis), the distinction between mainstream image analysis and CBIR is usually fairly clearly [12].

An example may make this clear. Many police forces now use automatic face recognition systems. Such systems may be used in one of two ways. Firstly, the image in front of the camera may be compared with a single individual's database record to verify his or her identity. In this case, only two images are matched, a process few observers would call CBIR. Secondly, the entire database may be searched to find the most closely matching images

III. EXISTING SYSTEM

A moving kNN query continuously reports the k nearest neighbors of a moving query point.
Location-based service providers (LBS) that offer remote kNN querying services often return mobile users a safe region to the query results. The main memory is always limited and the distance matrix could expand to an enormous size, a compact data structure that consumes a user-defined portion of main memory.

The basic task of D-cache is determine tight lower- and upper bound of an unknown distance between two objects, based on stored distances computed during previous querying and/or indexing.

IV. PROPOSED SYSTEM

We introduce a new extraction approach with caching distance. This is called as a D-cache. User entered a distance range search and find out the results. Here we are going to use parsing technique. it can extract the results from desired caches and distances. Hence, it will give the faster extraction results. Due to our proposed approach, the main advantages are high performance, low computational cost and low processing time.

V. D CACHE

Main concept about this paper is D Cache. D Cache is a technique/tool for general metric access methods that helps to reduce a cost of both indexing and querying. The main task of D cache is to determine tight lower and upper bound of an unknown distance between two objects. First we have to understand about Metric access methods— Metric access methods are the technique which is used in that situation where the similarity searching can be applied. E.g. search for SBI, it can search in entire country i.e. similar search has been invoked. First try to understand the concept of similarity searching. When any user submits a query in the search box or any database then the process of responding to these queries is termed as similarity searching. Given a query object this involves finding objects that are similar to q in a database S of N objects, based on some similarity measure. Both q and s are drawn from some "universe" U of objects, but q is generally not in S. We assume that the similarity measure can be expressed as a distance metric such that d(01,02) becomes smaller as 01 and 02 are more similar thus (s, d) is said to be a finite metric space.

Now, metric access method will facilitate the retrieval process by building an index on the various features which are analogous to attributes. These indexes are based on treating the records as a points in a multidimensional space and use point access methods. Metric access methods uses a structure for caching distances computed during the current runtime session. The distance cache ought to be an analogy to the classic disk cache widely used in DBMS to optimize I/O cost. Hence instead of sparing I/O, the distance cache should spare distance computations.

The main idea behind the distance caching resides in approximating the requested distances by providing their lower and upper bound. In whole project, there are mainly two operations used for both side i.e. for administrator and user. Each operation is worked by different algorithm.

A. Distance Calculation**:**

Distance Calculation operation is performed on user side. When any user types any keyword then distance will be calculated. The main D Cache functionality is operated

by methods(get distance, get lower bound distance)that means while distance retrieval process, first distance will be found and lower bound *distance* will be first allocated. For that purpose **"Algorithm for Distance Calculation"** is used.

The number of dynamic pivots used to evaluate get lower bound distance which is set by the user while this parameter is an exact analogy to the number of pivots in pivot tables. First we should try to understand about pivot tables and M

Tree.

B. M Tree:

The M Tree is a dynamic index structure that provides good performance in secondary memory.

The M Tree is a hierarchical index, where some of the data objects are selected as centers (local pivots) of ball shaped regions, while the remaining objects are partitioned among the regions in order to build up a balanced and compact hierarchy of data regions. So, with the help of pivot tables and M Tree construction, Distance is retrieved.

VI. EXPERIMENTAL STRATEGY

There are 6 Module in our Project

- Parent Filtering.
- NN Graph Filtering.
- Nearest Neighbor Graph.
- Range Query.
- M-Tree Querying Cost.
- M-Tree Constructing Cost.

A. Parent Filtering:

The tree contains the distances from the routing/ground entries to the center of its parent user; some of the non-relevant M-tree branches can be filtered out without the need of a distance computation.

The tree of parent filtering is |_ (P,Q) − _(P,R)| > rQ + rR, the data ball R cannot overlap the query ball, thus the child user has not to be re-checked by basic filtering.

The Sequential file was simplify by the query is (P,Q) was computed in the previous (unsuccessful) parent's basic filtering.

B. NN Graph Filtering:

The filtering using NN-graph is similar to the parent filtering, however, instead of the parent; we use an object S from the node.

The query selects such object S; then its distance to the query object Q is explicitly computed. The object S a sacrifice pivot, since to rescue other objects from basic filtering, this one must be sacrificed to the NN graph Filtering.

C. Nearest Neighbour Graph:

The Nearest Neighbour Process is same to M-tree Construction Process

The original M-tree proposal, the index was constructed by multiple dynamic insertions, which consisted of two steps.

First Step is, the leaf node for the newly inserted object is found by traversing a single path in the tree.

Second Step is, The leaf gets overfull after the insertion, it is split, such that two objects from the split leaf are selected as centre of the new two leafs, while the remaining objects within the split leaf are distributed among the new leafs.

D. Range Query:

The implementing a query processing, the tree structure is traversed such that non-overlapping users are excluded from further processing.

The basic and parent filtering, in M*-tree we can use the NN-graph filtering step (inserted after the step of parent filtering and before the basic filtering),while we hope some distance computations needed by basic filtering after an unsuccessful parent filtering will be saved.

The kNN algorithm is a bit more difficult, since the query radius rQ is not known at the beginning of kNN search of NN filtering user.

The listing of kNN pseudo code, however, its form can be easily derived from the original M-tree's kNN algorithm and the M*-tree's range query implementation presented above.

E. M-Tree Querying Cost:

The range search is always more efficient in M*-tree than in M-tree, because only such entries are chosen as to filtered by the parent, so for them distance computation is unavoidable.

The NN-graph filtering some of the entries can be filtered before they become a sacrifice, thus distance computations are reduced in this case.

F. M-Tree Constructing Cost:

M-tree, the navigation to the target leaf makes use of NN-graph filtering, so we achieve faster navigation.

The M-Tree insertion into the leaf itself the update of leaf's-graph is needed, which takes m distance computations for M-tree instead of no computation for M-tree.

The expensive splitting of a node does not require any additional distance computation, since all pair wise distances have to be computed to partition the node, regardless of using M-tree or M-tree

VII. CONCLUSION

In this article we presented the D-cache, a main-memory data structure which tracks computed distances while inserting objects or performing similarity queries in the metric space model. Since distance computations stored in the D-cache may be reused in further database operations, it is not necessary to compute them again. Also, the D-cache can be used to estimate distance functions between new objects and objects stored in the database, which can also avoid expensive distance computations.

ACKNOWLEDGEMENT

I express great many thanks to all faculties for supervising and leading me, to accomplish this fine work. To my family, for their warm, kind encourages and loves. To every person who gave me something too light along my pathway. I thanks for believing in me.

REFERENCES

[1] J.S. Vitter, "External Memory Algorithms and Data Structures: Dealing with Massive Data," ACM Computing Surveys, vol. 33, no. 2, pp. 209-271, citeseer.ist.psu.edu/vitter01external.html, 2001.

[2] C. Bo¨hm, S. Berchtold, and D. Keim, "Searching in High- Dimensional Spaces—Index Structures for Improving the Performance of Multimedia Databases," ACM Computing Surveys, vol. 33, no. 3, pp. 322-373, 2001.

[3] S.D. Carson, "A System for Adaptive Disk Rearrangement," Software—Practice and Experience, vol. 20, no. 3, pp. 225-242, 1990.

[4] W. Effelsberg and T. Haerder, "Principles of Database Buffer Management," ACM Trans. Database Systems, vol. 9, no. 4, pp. 560- 595, 1984.

[5] M. Batko, D. Novak, F. Falchi, and P. Zezula, Scalability Comparison of Peer-to-Peer Similarity Search Structures," Future Generation Computer Systems, vol. 24, no. 8, pp. 834-848, 2008.

[6] P. Zezula, G. Amato, V. Dohnal, and M. Batko, Similarity Search: The Metric Space Approach (Advances in Database Systems). Springer, 2005.

[7] E. Cha´vez, G. Navarro, R. Baeza-Yates, and J.L. Marroquı´n, "Searching in Metric Spaces," ACM Computing Surveys, vol. 33, no. 3, pp. 273-321, 2001.

[8] G.R. Hjaltason and H. Samet, "Index-Driven Similarity Search in Metric Spaces," ACM Trans. Database Systems, vol. 28, no. 4, pp. 517-580, 2003.

[9] H. Samet, Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, 2006.

[10] T. Skopal and B. Bustos, "On Index-Free Similarity Search in Metric Spaces," Proc. 20th Int'l Conf. Database and Expert Systems Applications (DEXA '09), pp. 516-531, 2009.

[11] E. Vidal, "New Formulation and Improvements of the

Nearest-Neighbour Approximating and Eliminating Search

Algorithm (AESA)," Pattern Recognition Letters, vol. 15, no. 1, pp. 1-7, 1994.

[12] M.L. Mico´ , J. Oncina, and E. Vidal, "An Algorithm for Finding Nearest Neighbour in Constant Average Time with a Linear Space Complexity," Proc. Int'l Conf. Pattern Recognition, 1992.

