Distance Cache by Metric Access Methods: Survey

Sonali S Surawase[1], Madhav Ingle[2]
[1, 2] Dept of Computer
[1, 2] JSPM,Hadapsar

Abstract- *In this survey paper the metric access methods is used to increase the DBMS performance and resolve all issues and risks. The new caching techniques and buffering techniques used to consume the I/O cost utilization. The caching of accessed disk pages has been successfully used for decades in database technology, resulting in effective amortization of I/O operations needed within a stream of query or update requests. However, in modern complex databases, like multimedia databases, the I/O cost becomes a minor performance factor. In particular, metric access methods (MAMs), used for similarity search in complex unstructured data, have been designed to minimize rather the number of distance computations than I/O cost (when indexing or querying). Inspired by I/O caching in traditional databases, in this paper we introduce the idea of distance caching for usage with MAM. User forward the query, same query result is present in different databases; using similarity operation extracts the results from the distributed databases. Here utilization of I/O cost and CPU cost is high. It can have minor performance under computation cost.*

Keywords- Distance Cache, complex databases, indexing, database technology, Metric Access Method, M Tree

I. INTRODUCTION

Distance cache means a nonpersistent (main-memory) structure that stores distances already computed by a MAM. We consider a single runtime session of a search engine, that is, a contiguous usage of a MAM for a sequence of queries, insertions, or both. 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
The M Tree is a dynamic index structure that provide good performance in secondary memory.

The M Tree is a hierarchical index, where some of the data objects are selected as centres (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.

II. RELATED WORK

In this section, we survey previous work on metric access methods and pivot selection algorithms. A. Metric Access Methods Two broad categories of MAM exist, namely, compact partitioning methods and pivot-based methods. Compact partitioning methods partition the space as compact as possible and try to prune unqualified regions during search. BST [2], [3] is a binary tree built recursively. It uses a center with a covering radius to represent a partition. GHT [3] uses two centers for each tree node, and it divides the space according to which of the two centers is closer to every object. GANT [6] is an m-way generalization of GHT. It uses a Voronoi-like partitioning of the space, and a dynamic structure EGANT has also been proposed [3]. SAT [2] is also based on Voronoi diagrams, but unlike GHT and GNAT, it attempts to approximate the structure of a Delaunay graph. Dynamic and secondary memory extensions of SAT are also available [5], [3]. Next, the M-tree [5] is a height-balanced tree that is optimized for secondary memory. It is the first dynamic MAM, and it supports insertion and deletion. Several variants of M-trees, such as the Slim-tree [2], DBM-tree [4], and CM-tree [3], try to reduce the overlap among nodes and to further compact each partition. The D-index [1] is a multilevel structure that hashes objects into buckets, which are search-separable. LC [2] employs a list of clusters, which trades construction time for query time.

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 anun known distance between two

objects, based on stored distances computed during previous querying and/or indexing.

IV. PRPOSED SYSTEM

- The D-cache, a main memory data structure which tracks computed distances while inserting objects or performing similarity queries in the metric space Model.
- The D-cache 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.
- The experiments have shown the M-tree enhanced by NN-graphs can perform significantly faster, while keeping the construction cheap.

V. MODULES

Using 6 Models in our Project

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

MODULES DESCRIPTION

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.

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.

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.

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 centers of the new two leafs, while the remaining objects within the split leaf are distributed among the new leafs.

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 parentfiltering 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 notknown 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.

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 theybecome a sacrifice, thus distance computations are reduced in this case.

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 distancecomputation, since all pair wise distances have to be computed to partition the node, regardless of using M-tree or M*-tree.

VI. ACKNOWLEDGMENT

The the author gives special thanks to my parents, and department faculties for their great encouragement and cooperation in all aspects to develop this paper and to every person who gave me something too light along my pathway.

REFERENCES

[1] Lu Chen, YunjunGao, Xinhan Li, Christian S. Jensen †4, Gang Chen " Efficient Metric Indexing for Similarity Search," in IEEE ICDE Conference pp 591-601, 2015.
[2] Rajnish Kumar, PradeepBhaskar Salve "A Distance Cache Mining by Metric Access Methods" International Journal of Engineering Trends and Technology (IJETT) – Volume 4 Issue 9- Sep 2013.
[3] H. Zhao, W. Meng, Z. Wu, and C. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," Proc. 32nd Int'1 Conf. Very Large data Bases (VLDB), 2006.
[4] V. Crescenzi, P. Merialdo, and P. Missier, "Clustering Web Pages Based on Their Structure," Data and Knowledge Eng., vol.54, pp. 279-299, 2005.
[5] B. Liu, R.L. Grossman, and Y. Zhai, "Mining Data Records in Web Pages," Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 601-606, 2003.
[6] K. Simon and G. Lausen, "ViPER: Augmenting Automatic Information Extraction with Visual Perceptions," Proc. Conf. Information and Knowledge Management (CIKM), pp. 381-388, 2005.
[7] M. Wheatley, "Operation Clean Data", CIO Asia Magazine.
[8] N. Koudas, S. Sarawagi and D. Srivastava, "Record Linkage: Similarity Measures and Algorithms", Proc. ACM SIGMOD Int'1 Conf. Management of Data, pp. 802-803, 2006.
[9] R. Bell and F. Dravis, "Is You Data Dirty? and Does that Matter?," Accenture Whiter Paper, http://www.accenture.com, 2006.
[10] KomalSurawase, SonaliSurawase,AnujaPatil "Metric Access Methods for Distance Cache Mining" International Engineering Research Journal (IERJ) Volume 1, Issue 2, pp43-46, 2015, ISSN 2395-1621, 2015.
[11] P. Zezula, G. Amato, V. Dohnal, and M. Batko, Similarity Search: The Metric Space Approach (Advances in Database Systems). Springer, 2005