SCALABLE AND EFFICIENT CLUSTER-BASED FRAMEWORK FOR MULTIDIMENSIONAL INDEXING

Aretty Narayana[1], M. Srinivasa Rao[2], R.V.Krishnaiah[3]

[1]M.Tech Student, [2]Associate Professor, [3]Principal, Dept. of CSE, DRKCET, Hyderabad, AP, India
arettynarayan4u@gmail.com**, **msrao@drkgroup.org, r.v.krishnaiah@gmail.com

Abstract

Indexing high dimensional data has its utility in many real world applications. Especially the information retrieval process is dramatically improved. The existing techniques could overcome the problem of "Curse of Dimensionality" of high dimensional data sets by using a technique known as Vector Approximation-File which resulted in sub-optimal performance. When compared with VA-File clustering results in more compact data set as it uses inter-dimensional correlations. However, pruning of unwanted clusters is important. The existing pruning techniques are based on bounding rectangles, bounding hyper spheres have problems in NN search. To overcome this problem Ramaswamy and Rose proposed an approach known as adaptive cluster distance bounding for high dimensional indexing which also includes an efficient spatial filtering. In this paper we implement this high-dimensional indexing approach. We built a prototype application to for proof of concept. Experimental results are encouraging and the prototype can be used in real time applications.

Index Terms–*Clustering, high dimensional indexing, similarity measures, and multimedia databases*

--***--

1. INTRODUCTION

Data mining has been around for many years for programmatically analyzing huge amount of historical data. Clustering is one of the data mining techniques which is widely used to group similar objects into a clusters. However, clustering is not easy with high-dimensional data which is being produced in many domains such as digital multimedia, CAD/CAM systems, Geographical Information Systems (GIS), stock markets, and medical imaging. The high dimensional data is very huge ranging from 100 GB to 100 TB or more. Enterprises have to deal with such huge data. Such data is accessed using NN queries and spatial queries. Many researches came into existence on queries high dimensional data [1], [2]. These techniques use similarity measures like Euclidean Distance while making clustering. In [3] search performance has been improved using R-tree like structures. All the existing techniques expect the data sets to have uniform distribution of correlations in order to overcome the problem of "curse of dimensionality". As the nearest and distant neighbors are not distinguishable, indexing such datasets is not possible. The real world data sets on the other hand have non-uniform distribution of correlations. Therefore these data sets can be indexed really using techniques such as Euclidean Distance. The usage of ED is a significant research activity which is used in many applications including content – based image retrieval [4], [5]. In this paper we focused on the real world datasets which are high-dimensional in nature for indexing.

2. PRIOR WORKS

In case of low dimensional data techniques such as hyper-rectangles [6], [7], hyper spheres [8] or combination of both [9] were used for NN searches and effective data retrieval. Manyresearches were found in the literature to deal with high-dimensional data. Multi-dimensional indexing always outperforms low dimensional structures as they can access data quickly when compared with sequential scan. However, Weber et al. [10] proved that when dimensionality is more than 10, sequential scan is better than using indexing. The degradation of performance is due to the "curse of dimensionality" concept proposed in [11]. VA-files concept became popular to overcome thisproblem. The VA-File technique divides space into many hyper-rectangular cells and the approximation file holds strings pertaining to encoded bits that represent non-empty cell locations. The vector approximation ultimately leads to scalar quantization which overcomes curse of dimensionality. LDR [12] uses non-linear approximation in order to perform sequential scan. It is achieved by using dimensionality reduction and also clustering. There are some hybrid methods such as IQ-Tree [13] and A-Tree [14] that makes use of both VA and tree based index. Other methods like Pyramid Tree [15], iDistnce [16] and LDC [17] reduce transformations based on local dimensionality. A reference point is used to evaluate partitions made on dataset. Each partition has feature vectors that are indexed using their centroid-distance. When queries are made the spheres increase gradually until the cluster sphere is intersected. For quality reasons, the query processing identifies centroid – distances. The radius for search is adjusted in such a way that the NN query returns exact results.

Another experiment is made using approximation layer in LDC [17] by creating a box identification code which represents resident points. Once set of candidates are identified, then the B+ tree is used to filter out unwanted points. The resultant elements are finally retrieved in the NN result. Care is taken to control search space in order to reduce disk IO and also ensure that accurate results are retrieved by NN queries.

3. CLUSTER DISTANCE BOUNDING

Our procedure to measure distances to clusters is based on the cluster distance bounding proposed by Ramaswamy and Rose [18]. The architectural overview of the frameworkproposed by them is as shown in fig. 1.

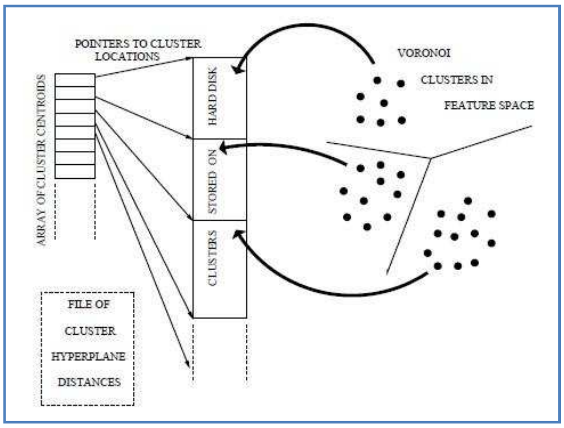

Fig. 1 –Architecture of Proposed Index Structure (excerpt from [18])

As seen in fig. 1the index structure contains array of cluster centroids which point to the cluster locations. The whole structure is like an index in book which will improve the query processing speed. In the same fashion, the NN queries and other queries on high-dimensional data can be processed faster. The data objects are grouped into number of clusters namely Voronoi clusters. The cluster-based indexing makes it more flexible and adaptable thus making the whole system scalable. This architecture makes the query retrieval very efficient and faster than other existing clustering algorithms and related indexes.

Their algorithms are used to implement the prototype which demonstrates the high dimensional indexing process. The first algorithm we used in the prototype application is to make voronoi clusters. The algorithms are presented in fig. 2, fig. 3 and fig. 4. More information about the algorithms and notations used inthe algorithms can be found in [18]. It is as shown in fig. 1

Algorithm 1 VORNOI-CLUSTERS(X, K)

1: // Generic clustering algorithm returns
// K cluster centroids
$\{c_m\}^K_{=1}$ ---- GenericCluster(X,K)
2: set l=0, $X_1 = Ø$, $X_2 = Ø$…… $X_K = Ø$
3: while l< |X| do
4: l=l+1
5: //Find the centroid nearest to data element X_l
K=arg min $d(X_l, C_m)$
6: //Move X_l to the corresponding Voronoi partition
$X_K = X_K$ Ụ $\{X_l\}$
7: end while
8: return $\{X_m{}^K{}_{=1}\}$, $\{c_m\}^K_{=1}$

Fig. 2–Algorithm for Voronoi Clusters (excerpt from [18])

As can be seen in fig. 2, the algorithm takes a dataset and K value as input and generates and returns required Voronoi clusters. Initially a generic clustering algorithm like K-means is used to get cluster centroids. The algorithm further processes and finally returns Voronoi clusters. In fig. 3 algorithm 2 is presented. It is meant for making kNN search.

Algorithm 2 KNN-SEARCH(q)

1: //Initialize
set FLAG=0, count = 0,N = 0, kNN = φ
2: //Evaluate query-cluster distance bounds
$d_{LB}[] \leftarrow$HyperplaneBound(q)
3: //Sort the query-cluster distance bounds in ascending
//order
$\{d^{sort}{}_{LB}[], o[]\} \leftarrow$SortArray($d_{LB}$,'ascend')
4: while FLAG==0 do
5: count = count + 1
6: //Find the kNNs upto current cluster
$\{Nc, kNN\} \leftarrow$FindkNNsIn(q,$X_{o[count],}$ kNN)
7: //Update number of elements scanned
$N = N + N_c$
8: //Find the kNN radius
d_{kNN}=Farthest(q, kNN)
9: if count < K then
10: if N >k then
11: if $d_{kNN} < d^{sort}{}_{LB\ [count + 1]}$ then
12: set FLAG=1 //kNNs found, search ends

```
13: end if
14: end if
15: else
16: set FLAG=1 //all clusters scanned, search ends
17: end if
18: end while
19: return kNN
```

Fig. 3 – kNN Search Algorithm

As seen in fig. 3, it is evident that the kNN algorithm is meant for performing k-Nearest Neighbor Search. It takes a query as input and returns k-Nearest Neighbors as possible search results. Algorithm 3 is meant for providing some support to algorithm 2.

Algorithm 3 FindkNNsIn(q,A, I)

```
1: set Nc= 0, F=Open(A), kNN = I
2: while !(EOF(F)) do
3: // Load the next cluster page
 C=LoadNextPage(F)
4: //Merge kNN list with current page
 Xcand = C Ụ kNN
5: //Find the kNNs within the candidate list
 kNN[] ←FindkNN(q,Xcand)
6: //Update number of elements scanned
Nc= Nc+ |C|
7: end while
8: return Nc, kNN
```

Fig. 4–Algorithm for some intermediary functionality

As seen in fig. 4, it is evident that it performs some functionality and returns intermediary results which are used by its caller. Its caller is algorithm 2 that is meant for k-NN search which returns k-Nearest Neighbors that satisfy the given query.

4. RESULTS

We built a prototype application to demonstrate the proof of concept of the paper. We have used many real time data sets to apply the index structure proposed in [18]. The datasets used in experiments include SENSORS, HISTOGRAM, AERIAL, CORTINA and BIO-RETINA. The details of dimensions, number of vectors and size in terms of pages of these data sets are presented in table 1.

Table 1- Dataset details

Name	Dimensionality	No. of Vectors	Size (Pages)
HISTOGRAM	64	12,103	379
SENSORS	60	50,000	1471
AERIAL	60	275,465	8300
BIO-RETINA	62	208,506	6200
CORTINA	74	1,088,864	40,329

As can be seen in table 1, the dimensionality of data sets and the number of vectors involved in each dataset are given. The environment used for developing prototype application includes a PC with 4GB RAM, Core 2 Dual processor running Windows XP operating system. Java platform is used to build the application. NetBeans is used as an IDE (Integrated Development Environment). Experiments are made in terms of number of sequential pages scanned, number of random IO operations, with KNNs and various datasets.

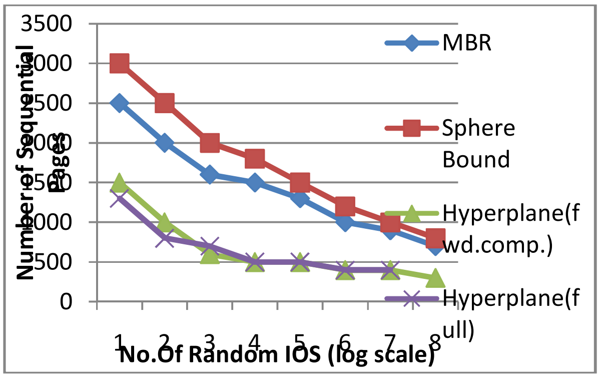

Fig. 5–IO Performance of Distance Bounds (BIO-RETINA dataset)

As can be seen in fig.5, the horizontal axis represents number of random IOs while the vertical axis represents number of sequential pages. As shown in results hyperlane bounds outperform the other bounds.

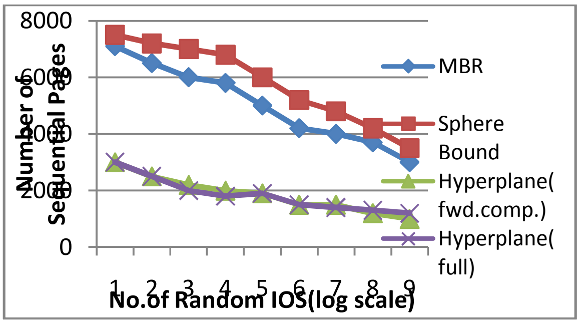

Fig. 6 – IO Performance of Distance Bounds (AERIAL dataset)

As can be seen in fig.6, the horizontal axis represents number of random IOs while the vertical axis represents number of sequential pages. As shown in results hyperlane bounds outperform the other bounds.

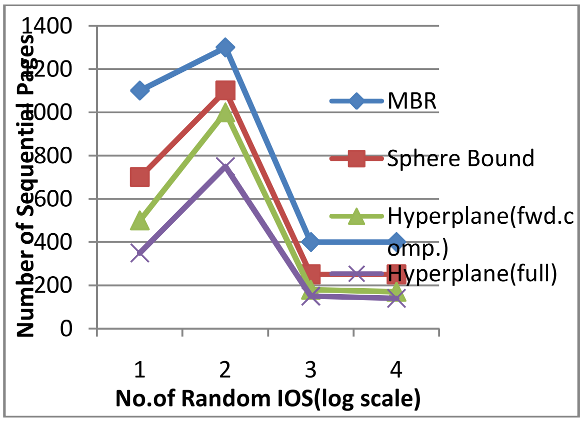

Fig. 7 – IO Performance of Distance Bounds (SENSORS dataset)

As can be seen in fig. 7, the horizontal axis represents number of random IOs while the vertical axis represents number of sequential pages. As shown in results hyperlane bounds outperform the other bounds.

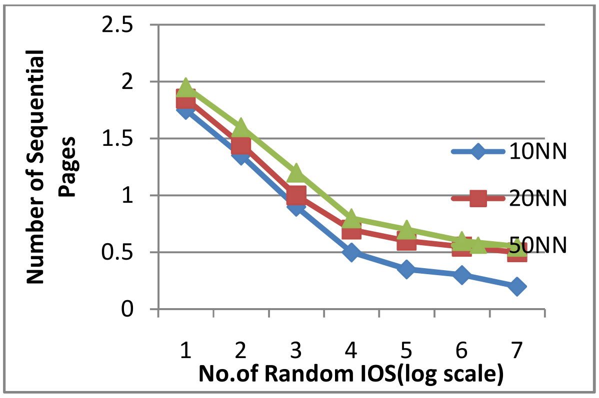

Fig. 8 – IO Performance of Distance Bounds (SENSORS dataset)

As can be seen in fig. 8, the horizontal axis represents number of random IOs while the vertical axis represents number of sequential pages. As shown in results hyperlane bounds outperform the other bounds.

CONCLUSIONS

Dealing with real multi-dimensional datasets that have correlations distributed in non-uniform fashion is not easy. Indexing such data is a challenging task. The existing indexing methods used VA-File through scalar quantization. However, this approach is proved to be suboptimal. Therefore, in this paper, we implement a new approach proposed by Ramaswamy and Rose [18] which groups datasets into Voronoi clusters. It is achieved by using similarity measures such as Mahalanobis and Euclidean. This has resulted in the reduction of IOs. We also built a prototype application which demonstrates the utility of the proposed approach. We made experiments using various multi-dimensional data sets. The empirical results revealed that the indexing approach followed in the prototype is effective and can be used in the real world applications.

REFERENCES

[1] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is“nearest neighbor” meaningful?” in ICDT, 1999, pp. 217–235.
[2] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprisingbehavior of distance metrics in high dimensional spaces,” in ICDT, 2001,pp. 420–434.
[3] B. U. Pagel, F. Korn, and C. Faloutsos, “Deflating the dimensionalitycurse using multiple fractal dimensions,” in ICDE, 2000, pp. 589–598.
[4] T. Huang and X. S. Zhou, “Image retrieval with relevance feedback:From heuristic weight adjustment to optimal learning methods,” in ICIP,vol. 3, 2001, pp. 2–5.
[5] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, “Information-theoreticmetric learning,” in ICML, 2007, pp. 209–216.
[6] A. Guttman, “R-trees: A dynamic index structure for spatial searching.”in SIGMOD, 1984, pp. 47–57.
[7] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, “The R*-tree:an efficient and robust access method for points and rectangles,” inSIGMOD, 1990, pp. 322–331.
[8] D. A. White and R. Jain, “Similarity indexing with the SS-tree,” inICDE, 1996, pp. 516–523.
[9] N. Katayama and S. Satoh, “The SR-tree: An index structure for highdimensionalnearest neighbor queries.” in SIGMOD, May 1997, pp. 369–380.
[10] R. Weber, H. Schek, and S. Blott, “A quantitative analysis and performancestudy for similarity-search methods in high-dimensional spaces.”in VLDB, August 1998, pp. 194–205.
[11] R. Bellman, Adaptive Control Processes: A Guided Tour. Princeton,NJ: Princeton University Press, 1961.
[12] K. Chakrabarti and S. Mehrotra, “Local dimensionality reduction: A newapproach to indexing high dimensional spaces.” in VLDB, September2000, pp. 89–100.
[13] S. Berchtold, C. Bohm, H. V. Jagadish, H. P. Kriegel, and J. Sander,“Independent Quantization: An index compression technique for highdimensionaldata spaces,” in ICDE, 2000, pp. 577–588.
[14] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima, “The A-tree: Anindex structure for high-dimensional spaces using relative approximation,”in VLDB, September 2000, pp. 516–526.

[15] S. Berchtold, C. Bohm, and H. Kriegel, “The Pyramid-technique:Towards breaking the curse of dimensionality,” in SIGMOD, 1998, pp.142–153.
[16] C. Yu, B. C. Ooi, K. L. Tan, and H. V. Jagadish, “Indexing the distance:An efficient method to knn processing.” in VLDB, September 2001, pp.421–430.
[17] N. Koudas, B. C. Ooi, H. T. Shen, and A. K. H. Tung, “LDC: Enablingsearch by partial distance in a hyper-dimensional space.” in ICDE, 2004,pp. 6–17.
[18] Sharadh Ramaswamyand Kenneth Rose, “Adaptive Cluster Distance Bounding for High Dimensional Indexing”. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL.23, NO. 6, June 2011. P1-15.

BIOGRAPHIES:

Aretty Narayana has completed B.Tech (E.C.E) from Srinivas Reddy Institute of Technology and pursuing M.Tech (C.S.E) in DRK College of Engineering and Technology, JNTUH, Hyderabad, Andhra Pradesh, India. His main research interest includes Data Mining, Database and DWH.

M. Srinivasa Rao is working as an Associate Professor in DRK College of Engineering and Technology, JNTUH, Hyderabad, Andhra Pradesh, India. He is pursuing Ph.D in Information Security. He has completed M.Tech (C.S.E) from JNTUH. His main research interest includes Information Security and Computer Ad-Hoc Networks.

Dr. R. V. Krishnaiah, did M.Tech (EIE) from NIT Waranagal, MTech (CSE) form JNTU, Ph.D, from JNTU Ananthapur, He has memberships in professional bodies MIE, MIETE, MISTE. His main research interests include Image Processing, Security systems, Sensors, Intelligent Systems, Computer networks, Data mining, Software Engineering, network protection and security control. He has published many papers and Editorial Member and Reviewer for some national and international journals.