World Engineering & Applied Sciences Journal 8 (2): 57-65, 2017
ISSN 2079-2204

DOI: 10.5829/idosi.weasj.2017.57.65

Distributed Weighted Fuzzy C-Means Clustering Method with Encoding Based Search Structure for Large Multidimensional Dynamic Indexes

[1]Sunil Sunny Chalakkal, [2]M. Rajalakshmi and [3]R. Vijayakumar

[1]Department of Computer Science, St Thomas College, Trichur, Kerala, India
[2]Department of CSE / IT, Coimbatore Institute of Technology, Coimbatore, India
[3]School of Computer Science, M.G.University, Kottayam, India

Abstract: Searching of data in huge dimensional space is a major issue now days. In this paper different index structure and improved cluster Tree++ structures are used to speed up the search process. With the help of high dimensional spaces with Distributed Weighted Fuzzy C-Means (DWFCM) clustering algorithm. All data points are grouped into clusters through a DWFCM clustering algorithm. Dual –distance –Driven encoding (DDE) approach is used to acquire the uniform ID number of every data point. Where every cluster is partitioned in to two based on the centroid distance. Finally, a uniform index key is obtained by integrating the unique ID number and centroid displacement. In addition, encoding based indexing approach is added to the improved Cluster Tree++ for distributed multidimensional data point. Evaluation on the effectiveness and efficiency of the proposed scheme is done through effective assessments, correspondingly the results states that this method is much better than the existing algorithms of Distributed Weighted Possibility-Means (DWPCM) and Weighted Possiblistic C-Means (WPCM).

Key words: Clustering · Indexing · Encoding · Fuzzy c-means · Index key · Dual-distance-Driven · B+-tree · Cluster Tree++ · distributed Multi-dimensional data sets

INTRODUCTION

Latest review states, large volumes of data with high dimensionality are generated in different fields. A lot of methods have been proposed for effectively indexing high-dimensional datasets for organized querying. When dimensionality scales up higher, the performance reduces with the present methods which can support nearest neighbor search for low dimensional datasets. Along with it, the process of adding new data can lead original structures to no longer maintain the dataset effectively, since it might considerably increase the amount of data accessed for a particular query. Amongst all these methods, Cluster Tree [1] is the startup work towards the direction of creating an efficient index structure from clustering for high dimensional datasets. An analysis scheme for finding out the interesting data distributions and patterns in the underlying dataset is termed as clustering. Given a set of *n*data points in a *d* – dimensional space, the clustering method provides the data points to groups in relation to the computation of the degree of similarity among data points in a group are extremely similar to each other compared to the data points in different groups. Each and every group is a cluster. In order to help well-organized queries the Cluster Tree scheme constructs an index structure on the cluster structures.

At the present time, a huge part of the data set is time associated; in addition to it availability of the outdated or unnecessary data in the data set shall critically degrade the dataset processing. On the other hand, few schemes are formulated in order to device the management of the complete data set to keep away from the outdated data and maintain the dataset constantly in a better and updated position for the easiness and effectiveness of dataset process such as query and insertion. Without having to linearly search for the high-dimensional dataset The Cluster Tree can support the retrieval of the

Corresponding Author: Sunil Sunny Chalakkal, Department of Computer Science, St Thomas College, Trichur, Kerala, India.

neighbors effectively. The key aim is to reduce the response time to the entire users query. It is the first scheme that uses the Cluster Tree for the purpose of generating effective index Structure for high dimensional datasets.

In this paper, a symmetrical encoding method is created by double partitioning. To deliver a key division of the uniform index key the uniform ID number of each point can be gained. In order to facilitate highly efficient DWFCM search a symmetrical encoding–based Improved cluster Tree++ (EIC-Tree++) is proposed. Using a DWFCM clustering algorithm all data are partitioned in to different clusters. Where all cluster spheres is segmented twice in relation with the two distances of start and centroid distance.

Section 2 reviews the related work cluster tree and index structure designs and clustering methods. Section 3 describes innovative symmetrical encoding-based index structure with clustering approach suitable to distributed multidimensional datasets. Section 4 focuses on the processing of Cluster Tree results and section 5 presents the conclusion and future research progress.

Related Work: Different types of schemes are being framed in order to manage effectively with multidimensional data. In particular, Space Filling Curve (SFC) schemes take part an important role. Space Filling Curve (SFC) methods include Z-order curve [2], Hilbert Curve [3] and gray codes [4] where data is divided onto multidimensional pixels in relation with the bottom granularity and uses a curve that passes through all the pixels in multidimensional space. An order of pixels in space is generated by this curve. This method (ordering) permits usage existing one dimensional access structures. For example, The B++Trees. Data representation on a part of the curve is enabled by the leaf pages of the access structure, generating a primary index where close by data are clustered with a high probability. The key drawback of the SFC scheme is that the CPU utilization rate is very high and they have issues such as high overlap among pages and the query interval.

For multidimensional data The UB-Tree [5] integrates a space filling curve and a B+-Tree generates the primary index. Each leaf node indicates a block of data on a part of the curve since it's a kind of paginated index. It divides the space into linear segments of a Z –Curve(or any SFC).The problem of the UB-Tree is that it requires amendments to the DBMS kernel for the purpose of integration and like other SFC's the partitions are not naturally not hyper –cubic and they may even represent disjoint space. The K-D-Tree is the most famous d-dimensional point data structures and its variants are: The hB-Tree [6], the BD-Tree [7], the hybrid tree [8] and the quad tree.

The binary search tree model includes The K-D –Tree which initiates a recursive subdivision of the data spaceto partitions through (d-1)-dimensional hyper planes.The most common problem to all the K-D –schemes is that for some distributions hyper planes that partitions the data objects equally are not found. Using the is-oriented hyper planes the K-D-Tree, the quad-tree [9] decomposes the universe. A major difference is that the quad-trees are not binary trees .Until the amount of objects in every partition is under a given threshold the subspaces are the subspaces are disintegrated. Quad trees are not stabilized and all the sub trees of populated trees need to deeper than sparingly populated areas leading a bad worst case behavior [10].

Pyramid Technique [11] is the transformation –based high dimensional indexing method. Works better incase of window based queries, here NB-tree [12, 13] and idistance [14] are used to manage the search process. For example NB-Tree is a kind of single reference point –based method. A B+-tree is used for indexing the distances on which all the operations are carried out.

The drawback of NB-Tree is that its inability to reduce the region of search, when the volume of data increase as well as the dimensionality shoot-up it will directly affect the searching process, the efficiency also decrees. iDistance [14] is framed by selecting certain reference points with the intention of further pruning. To increase the efficiency we are using the M-Tree [10], Omni-family [12] and empirically [14]. The efficiency of data searching is mainly depends on the portion mechanism, segmenting and clustering.

Proposed Clustering Based Indexing Structure: In this section, the proposed clustering based indexed structure is explained.The purpose of clustering the similar data points is done by the Distributed Weighted Fuzzy C-Means (DWFCM) algorithm which includes encoding based indexing scheme employed for serving search query results are detailed.

The major concept of Clustering-based indexing structures initially make use of clustering schemes with the idea of clustering data points and make use of estimation during search phase in such a way that the search shall be completed in the obtained clusters which likely includes the nearest neighbor of the query point. The architecture proposed for clustering –based structures is shown in the below figure.1 which includes two phases the clustering phase and the search phase.

The proposed architecture of the clustering-based structures is shown in the following Fig. 1 Clustering-based indexing structures includes two phases: clustering and search phase.

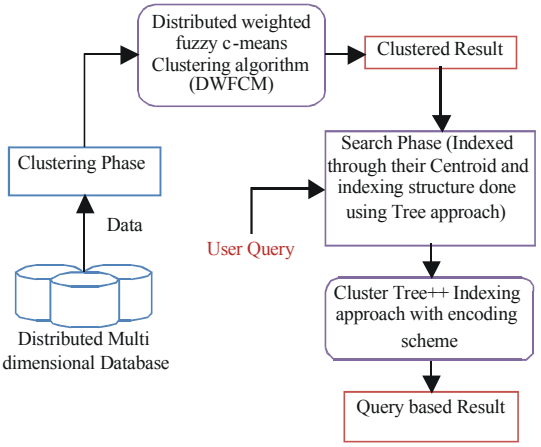

Fig. 1: Overall architecture diagram of proposed system

Clustering Phase: In this section, a Distributed WFCM algorithm (DWFCM) is formulated in accordance with Map Reduce in this section. Based on the steps of DWFCM, there are two most important operations: computing the degree of membership μ_{ij} and computing the clustering centers v_i. In the map phase, the Map function is intended to compute the degree of membership μ_{ij}.

Loading a huge amount of data in to memory makes some problems as same as in [15]. In the case only a particular amount of data is loaded in to memory. Suppose we are loading 1% of data into memory, we have to search 100 times scan to complete the entire search process. This method is called partial data access (PDA). The volume of data loaded for a given time will decide the amount of partial data access. In this method subsequent to the first PDA, Using Fuzzy C Means (FCM) data is grouped in to different clusters and reduced its weighted points. The additional mechanism we are going to implement is PDA. PDA is directly connected with weighted points. The major difference between the crisp clustering [15] scenario and this method is the absence of fuzzy membership. For every PDA, new singleton points are added into memory. This process will continue till the data is searched ones. The function of FCM is to accommodate weights. To get accurate clustering Result each data set will be closely connecting with centroids. This process of knowledge transmission permits for faster merging.

Weighted Point Calculation:

The FCM is defined as;

$$j_m(u,v) = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^{m} d_{ik}(x_k, v_i) \quad (1)$$

Calculation can be done using

$$u_{ik} = \frac{d_{ik}(x_k, v_i)^{\frac{1}{1-m}}}{\sum_{j=1}^{c} D_{jk}(x_k, v_j)^{\frac{1}{1-m}}} \quad (2)$$

v_i Presents the i^{th} cluster centroid and n presents the amount of examples and c indicates the volume of clusters. $d_{ik}(x_k, v_i) = \| x_k - v_i \|^2$ Presents the rule.At this stage the Euclidean distance method is used. A weighted FCM algorithm is taken into account which takes the weight of examples, clusters data into c partitions. Using n_d examples examples which are loaded in memory in the d_{th} as partial data access (PDA).

Case 1: d=1: When the value of d is equitable to one, in the first PDA, in this situation, As FCM WFCM is identical. Successively after clustering, consider v_i indicate the cluster centroids, where $1 \le i \le c$. Considering U_{ij} shows the membership values, where $1 \le i \le c$ and $1 \le i \le n_d$. Consider $\vec{w}$ shows the weights of the points in the memory. In such a situation, all the entire n_d points have weight 1 since weighted points from previous PDC do not exist.At this point of time the memory is let free through reducing the clustering result in to c weighted points, which are indicated by the c cluster centroids v_i, where $1 \le i \le c$ and their weights are shown below:

$$w_i' = \sum_{j=1}^{n_d} (U_{ij}) w_j, 1 \le i \le c \quad (3)$$

$$w_j = 1, \forall 1 \le j \le n_d \quad (4)$$

Weights of the c points, after the process of reduction the clustering results, in memory is shown below: followingthe process of condensing the clusteringresults, in memory is as given below:

$$w_i = w_i', 1 \le i \le c \quad (5)$$

It is observed that when n^d, In all the succeeding PDA ($d > 1$) new singleton points are added. Their indices related with $\vec{w}$ shall begin at $c + 1$ and finally end at $n_d + c$ i.e.

$$w_j = 1, \forall c < j \leq n_d + c \quad (6)$$

Case 2: $d > 1$: In this case, On singleton points clustering will be applied by newly loading in the d^{th} PDA along with $c$weighted points following the condensation from $(d-1)^{th}$ PDC. Resulting in $n_d + c$ points in the memory for clustering by means of WFCM. The derived new n_d singleton points have weight of one.After the clustering process, the data in the memory (togethersingletons and weighted points) is reduced to c new weighted points. All the new weighted points are indicated through the c cluster v_i, where $1 \leq i \leq c$ and their weights are formulated as below:

$$w_i^{'} = \sum_{j=1}^{n_d+c} (U_{ij}) w_j, 1 \leq i \leq c \quad (7)$$

Subsequently the memory is going to free, we can revise with this equation:

$$w_i = w_i^{'}, 1 \leq i \leq c \quad (8)$$

Weighted Fuzzy C Mean: To take into consideration of the weighted points the main objective of FCM is modified [16]. Each clustering group should have the c weighted point to accomplish perfect PDC. The closed centroids and the WFCM calculated as follows:

$$v_i = \frac{\sum_{j=1}^{n_d+c} w_j (U_{ij})^m x_j^{'}}{\sum_{j=1}^{n_d+c} w_j (U_{ij})^m}, 1 \leq i \leq c, x_j^{'} \in X^{'} \quad (9)$$

For each n_d we are assigning a value of one.

$$U_{ij} = \left[\sum_{i=1}^{c} \left(\frac{\|x_j^{'} - v_i\|}{\|x_j^{'} - v_l\|} \right)^{\frac{2}{m-1}} \right]^{-1}, 1 \leq i \leq c \text{ and } 1 \leq j \leq n_d + c \quad (10)$$

With the focus of reviewing cluster with the objective of cluster centers in parallel, two parameters, $\xi_i^{(t)}$ and $\lambda_i^{(t)}$ is introduced, where t shows the serial number of data node. Following After the process of calculating the membership μ_{ij}, Map function calculates $\xi_i^{(t)}$ and $\lambda_i^{(t)}$ by using equation (11) and (12);

$$\xi_i^{(t)} = \sum_{k=1}^{n/p} w_k u_{ik}^m x_k \; i = 1,2 \ldots c \quad (11)$$

$$\lambda_i^{(t)} = \sum_{k=1}^{n/p} w_k u_{ik}^m \; i = 1,2 \ldots c \quad (12)$$

For the purpose of the encoding scheme data point clustering results are taken. In this method, uniform, ID number of all data point is taken with the help of a encoding mechanism. with the help of unique ID and distance from the centroid, we create unique index key B+tree.

Symmetrical Dual Distance Encoding Scheme: Many observations has led to the initiation on design of improved cluster tree++. First we will measure the unlikeness of the data points in association with theire distance. Using the value we can create B+tree structure. To formulate the improved version of cluster tree++ we can combine two distance metrics. Through the use of a novel symmetrical encoding approach for the purpose of getting a uniform index key expression, to further diminish the search region.

Distance function, shown as $d(V_i, V_j)$ starting distance is the distance between it and $V_o(0,0,..0)$, naturally given as;

$$SD(V_i) = d(V_i, V_o)$$

It is observed that n data points are clustered into T clusters, the O_j of each and every cluster C_j is derived in the beginning, where $j \in [1,T]$.

For calculating centroid distance, V_i is the data point and distance from the centriod is O_j.

$$CD(V_i) = d(V_i, O_j)$$

where $i \in [1,\| C\|]$ and $j \in [1, T]$.

Dual distance encoding scheme DDE is used to get a uniform index key by means of double "slicing" mechanism. With the assistance of a DWFCM clustering algorithm, the start and centroid distances of every data point is computed, as a result V_i can be designed as a four tuple:

$$V_i :: = < i, CID, SD, CD >$$

Here i indicates to the i^{th} data pointand CID shows the cluster id.

Start Slice: Fora given cluster sphere (O_j, CR_j) the γ^{th} start slice of it is shown as $SS(\gamma, j)$ centroid slice is a cluster sphere (O_j, CR_j) where $\mu \in [1, \beta]$ here,

$V_i :: = < i, CID, SD\_ID, CD\_ID >$

In the given equation i is representing the ID number Vi, CID is the cluster ID, SD-ID means the starting slice ID number and CD-ID is the centoid distance ID number. From that we can form the equation given below;

$$\begin{cases} SD_{ID(V_i)} = \frac{\alpha}{2}\left[\frac{|SD(V_i) - SD(O_j)|}{2CR_j/\alpha}\right] \\ CD_{ID(V_i)} = 1 + \left[\frac{CR_j - CD(V_i)}{CR_j/\beta}\right] \end{cases} \quad (13)$$

when α shows an even integer and $\alpha/2+1= \beta$.

This results, a uniform ID number of V_i which can be indicated as linearly integrating theSD_ID and CD_ID as shown below:

$UID(V_i) = c \times SD\_ID(V_i) + CD\_ID(V_i)$

In relation to the encoding scheme , the uniform encoding identifier of each data point can be obtained through Algorithm 1.

Algorithm 1.
Problem:
Calculate the points DDE *(1 to n) from the set* Ω, Starting cluster is *α, the centroid slice is* β.

Steps:
1. Using *DWFCM* clustering algorithm, the data points in the set Ω are grouped into *T* number of clusters.
2. Set j=1
3. Repeat steps 4 through 9 until j<=T
4. Repeat steps 5 through 8 while $V_i \in \Theta(O_j, CR_j)$
5. Compute the centroid distance and start distance of each point.
6. finding out the result of the Eq.(13)
7. construct the unique ID of V_i ($DDE(V_i)$).
8. Take next V_i
9. Increment j by one
8. End

The data structure is given in relation with the above algorithm.With the above definitions, obtain the index key of V_i as given below:

$$key(v_i) = UID(V_i) + \frac{CD(V_i)}{MCD} = c \times \left(\frac{\alpha}{2} - \left[\frac{SD(V_i) - SD(O_j)}{\frac{2CR_j}{\alpha}}\right]\right) + 1 + \left[\frac{CR_j - CD(V_i)}{CR_j/\beta}\right] + \frac{CD(V_i)}{MCD} \quad (14)$$

The above Eq.(14) shows the index key expression of V_i. On the other side, in relation with the symmetrical encoding scheme, there exists a chance of availability f two different data points (*e.g.,* V_i and V_j) within the identical cluster sphere, where its index keys are equal. Both the two data points must meet the condition that: $(SD(O_j) - SD(V_i)) \times (SD(O_j) - SD(V_j)) < 0$.

In order to keep away from the overlapping of the index we using the equation below:

$$Ukey(V_i) = \begin{cases} SCALE_1 + key(V_i), if\ SD(V_i) \in [SD(O_j), SD(O_j) + CR_j] \\ SCALE_2 + key(V_i), if\ SD(V_i) \in [SD(O_j) - CR_j, SD(O_j)] \end{cases} \quad (15)$$

Finally, n index keys obtained by means of Eq. (15) is inserted into a partitioned B+- tree, which will be discussed in the search phase.

Search Phase: In this section , the hierarchical structure of the improved cluster++ is presented. Then a scheme is presented for breaking up a cluster inosubclusters and the scheme for the purpose generating a clusterTree++ by means of decomposing a cluster into subclusters and a method for the purpose of building a ClusterTree++ by emans of decomposing clusters repeatedly. The enhanced Cluster Tree++ is used to encode for the purpose of reducing the dead space and searching time. The key advantage of this scheme is that the loading process goes rapidly because the output can be given sequentially and it makes only one pass over the data and no blocks are required to be rearranged. Benefits in the tree are loaded and then the blocks are 100% complete. Sequential loading generates a degree of spatial locality inside the file==>.

Seeking can be reduced. The key three conditions are: When blocks are split in the sequence set, a new separator has to be inserted into the index set. Then when blocks are combined in the sequence set, a separator has to be eliminated from the index set. When records are re-distributed among blocks in the sequence set, the value of separator in the index set has to be changed.

Construction: The building of improved Cluster Tree++is based on the generation of Cluster Tree++ and the construction of simple prefix B+ tree in parallel with encoding. Using a hierarchical structure and T sliced indexes. During the construction of ClusterTree++ all internal node and leaf node must set the ct (current time) and ht (historic time) as the current time as similar to the current time. Since the insertion time of the original data points are not known in the dataset, resulting in it is necessary to set the insertion times of the entire original data points as the current one. When the new data points are inserted into the structure, different times can be recorded into the structure. During the meantime, a leaf node is developed in simple prefix B+tree which includes the current time data. The entire L field of the entries in the leaf node of improved clusterTree ++ must be pointed to the created leaf node in simple prefix+ tree. The entire L field of the entries in the leaf node of improved Cluster Tree++ must be pointed to the created leaf node in simple prefix B+ tree.

Algorithm 2: Improved clustertree++ Index Construction Problem: Given the data point set Ω, the algorithm constructs the index *bt*(1 *to T*) for Improved clustertree++

Steps:

1. Using *DWFCM* clustering algorithm, the data points in the set μ are grouped into *T* number of clusters.
2. Set j = 1
3. Repeat steps 4 through 11 until j<=T
4. Create index header file, *bt*(*j*) ← *new tree File*()
5. Repeat steps 6 through 9 for each data point V_i in the *j*-th cluster
6. In this step we are calculating the centroid distance.
7. Using algorithm 1 obtain the ID number of *Vi.*
8. Set *Key* (V_i) = *TransValue*(V_i)
9. Insert it to B+-tree using the function *Binsert* (*Key*(V_i), *bt*(*j*))
10. Return *bt(j)*
11. Increment j by one
12. End

Processing of the ClusterTree: Insertion , Query and Deletion are the three main process in the ClusterTree++:

Insertion: For new incoming data point, its classified into one three categories: *Clusterpoints*: In a cluster inside a given threshold extremely near to certain data points*Close by points*: The data points which are neighbors to certain points in the clusters inside a specific threshold. *Random points*: are the data points which are far from all the clusters and cannot be bounded through any bounding sphere of the cluster Tree, or shall be included in the bounding spheres of the clustering during every level, but they do not have any nearest cluster points inside a particular threshold.As a result in agreement with the category of the new coming data point, the insertion algorithm of Cluster Tree ++ shall be applied repeatedly to put in data point to a specific leaf node of cluster Tree++along with the insertion time in the *T*(*time*) field of the new entry in the leaf node, at the same time including a new entry which includes the insertion time details into

the simple prefix B+tree. Subsequently, point the *L*(*link*) field of the new entry in the certain leaf node in ClusterTree++ to the new entry in the leaf node of simple prefix B+ tree and point the *L*(*link*) field of the new entry in the particular leaf node in simple prefix B+ tree to the new entry in the leaf node of ClusterTree++.

Query: The time-irrelevant queries and time-associated queries are the two categories of queries. The time-irrelevant queries comprise the range queries and *p* Nearest Neighbor queries. The time-associated queries comprise certain time period condition.For instance, certain clients possibly will request the nearby clients to a particular data point that are added into the structure, approximately the same time of that data point insertion. The original Cluster Tree query algorithm can be employed to resolve the previous category of queries. The second category of queries is solved by using the algorithm time-related-queries.

Deletion: Within several systems the superseded or unnecessary data must be removed once in a while. Managers might desire to remove the dataset that are included , during the past, or they desire to remove those data inserted at some stage in a certain period. We can use time -related-deletion algorithm to solve this problem.

In addition, they can just point out to the data system to automatically regulate itself. The ClusterTree+ can support such user specified deletions. Using the recursive automatic adjustment algorithm we can make new ClusterTree.

EXPERIMENTAL RESULTS AND DISCUSSION

Here, distributed multi dimensional dataset is taken as input. The real-world datasets are utilized for carrying out the tests: CAR comprises 2, 249, 727 road segments of California obtained from Tiger/Line datasets; (b) HYD comprises 40, 995, 718 line segments symbolizing China rivers and (c) TLK includes up to 157, 425, 887 points obtained from the China's elevation data.

Memory Usage: Given that DWFCM process data as amount of chunks calculated as the memory utilization of every chunk independently and get the largest value as the closing memory consumption for DWFCM. In view of the fact that the dataset is streaming in character, it is not necessary for DWFCM to access over one chunk at a time. Figure 2 shows the percentage of improvement in terms of memory consumption by proposed (DWFCM) as compared against the Baseline Algorithm (DWPCM) and (WPCM).

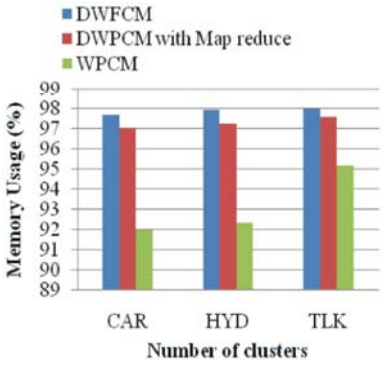

Fig. 2: Memory Usage Comparison

It is clear from Table 1, the improvement is in excess of 97% for the entire three datasets. WPCM makes use of the complete dataset at a time and that's why it needs adequate memory to hold the entire dataset. This is the cause why WPCM needs much higher memory than the proposed algorithm.

Table 1: Memory Usage Comparison

Input Dataset	DWFCM (%)	DWPCM with Map reduce (%)	WPCM (%)
CAR	97.7	97.05	92
HYD	97.95	97.25	92.35
TLK 98	97.62	95.21	

Selectivity: The time and the quantity of page writes for the purpose of bounding boxes insertion by means of several data chunks are illustrated in Figure 3. Since, only a predetermined size dataset undergoes partitioning, the size of the data chunk reduces, thereby increasing the amount of leaf nodes in the indexing trees. If the data chunk size is 300 × 204 (or 300 × 205), approximately 40, 000 data chunks are inserted into the indexing trees, however when the data chunk size is 20 × 204, approximately 560, 000 data chunks can be inserted.

As per fact, file pages accession reduces the response time to a definite query and increase the selectivity. Hence multi-dimensional indexing structures should supply low amount during the explore for accessing a file. It is clear from the table 2 that the proposed indexing scheme of DWFCM with cluster tree++ encoding is perform well than the DWPCM with cluster tree++ and WPCM with cluster tree+.

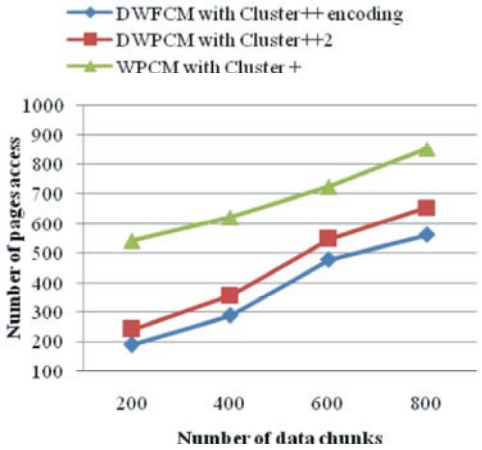

Fig. 3: Selectivity Comparison

Table 2: Selectivity Comparison

Number of Data Chunks	DWFCM with Cluster++ encoding	DWPCM with Cluster++2	WPCM with Cluster +
200	190	240	540
400	287	354	621
600	476	546	723
800	560	654	850

Scalability with Query: Figure 4 demonstrates the query experiment result. It is obvious from the results that Cluster Tree+ can resolve the multi-dimensional query inefficient setback; however Cluster Tree++ encoding execute much better than Cluster Tree++.

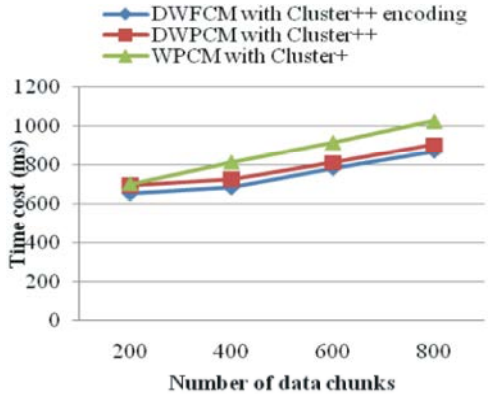

Fig. 4: Scalability Comparison

Table 3 shows the values of scalability of the methods. It proves that the multi-dimensional distributed index scales almost linearly with the number of nodes in the system.

Table 3: Scalability Comparison

Number of Data Chunks	DWFCM with Cluster++ encoding	DWPCM with Cluster++	WPCM with Cluster+
200	650	694	700
400	680	725	812
600	780	810	915
800	870	902	1025

Table 3 shows the values of scalability of the methods. With the number of nodes in the system, the multi-dimensional distributed index confirms the rank with approximate linear.

CONCLUSION

This paper presents a novel high-dimensional indexing in the midst of clustering scheme. This is phrased as symmetrical Encoding-based improved cluster-tree++ (EIC-Tree++). To build an EIC-Tree++, process carries out various steps. In the first step, with the support of a DWFCM algorithm, the complete (n) data points are clustered into T clusters. In the second step, through an undersized encoding effort, the uniform ID number of each point is gained, where every cluster sphere is segregated two times in accordance with the dual distances of start- and centroid-distance. Through the merging process of uique ID and centroid distanc Finally we getting the unique index key. The uniform index keys are through a partitioned B+-tree. Consequently, an advanced encoded ClusterTree++ will be capable of unvaryingly maintain the most renewed status of the dataset in order to sustain the competence and efficiency of data manipulations like insertion, modification and deletion. An advanced encoded ClusterTree++of hierarchy of clusters and sub clusters that integrates the representation of clusters into the index structure in order to accomplish the retrieval effectually and proficiently. The experimental outcomes reveal that the proposed method do one better than the existing Distributed Weighted Possibilistic C-Means (DWPCM) Algorithms and Weighted Possibilistic C-Means (WPCM) Algorithms. Future research must be concentrated on the storage and also must enhance the performance of the tree structure.

REFERENCES

1. Yu, Dantong and Aidong Zhang, 2000. Integration of Cluster representation and Nearest Neighbor search for Image Databases. IEEE international Conference on Multimedia, New York.
2. Orenstein and J.A. Merrett, 1984. A class of data structures for associative searching , Proceedings of 3rd ACM symposium on principles of database systems, New York.
3. Faloutsos, C. and S. Roseman, Fractals for secondary Key retrieval in the proceedings of the 8th ACM SIGACT-SIGMOD symposium on principles of data base.
4. Faloutsos, C., 1986. Mutihashing using gray codes, ”SIGMOD 86”Proceedings of the 1986 ACM international conference on Management of Data, NewYork, ACM Press, pp: 227-238.
5. Berchtold, S. and C. Bohm, 1999. Implementation of multidimensional index structures for knowledge discovery in relational databases-International conference on Data Warehousing ().
6. Lomet, D. and B. Salzberg, 1983. The Hb-Tree: A robust multiattribute search structure –Proceedings in IEEE international conference on Data engineering (1989) 296-304 Y. Ohsawa, M.S.: Bd-tree: A new n-dimensional data structure with efficient dynamic characteristics. Proceedings of the Ninth World Computer Congress, IFIP, pp: 539-544.
7. Chakrabarti, K., 1999. The Hybrid tree-An index structure for high dimensional feature spaces-Proceedings of the fifteenth international conference on Data Engineering, pp: 440-447.
8. Samet, H., 1984. The quadtree and related hierarchical data structures. ACM Computing Surveys 16(2): 187-260.
9. Ciaccia, P., M. Patella and P. Zezula, 1997. M-trees-Efficent access method for similarity search in metric space. Proceedings of the 23rd VLDB Conference, pp: 426-435.
10. Berchtold, S., C. Bohm and C. Bohm, 2001. The pyramid technique: towards breaking the curse of dimensionaliy. Proceedings of SIGMOD conference.
11. Filho, R., A. Traina and C. Faloutsos, Similarity search without tears-The omni family of all purpose access methods.In proceedings of ICDE Conference, pp: 623-630.
12. Fonseca, M.J. and J.A. George, 2003. Indexing High-Dimensional data for Content –Based retrieval in large databases.In proceedings of the DASSFA Conference, Kyoto, Japan, pp: 267-274.
13. Jagadish, H.V., B.C. Ooi and K.L. Tani, 2005. Distance: An adaptive B+-Tree based Indexing Method for Nearest Neighbor search, ACM Transactions on Database Systems, pp: 364-397.
14. Fredrik Farnstorm, Jameslewis and Charles Elkan, 2000. Scalability for Clustering algorithms revisisted, ACM SIGKDD Explorations, 2: 51-57.
15. Steven Eschrich, Jingweike Lawrence, O. Hall and B. Dmitry, 2003. Fast accurate Fuzzy Clustering through data reduction, IEEE Transactions on Fuzzy Systems, 11: 262-270.