International Journal of Computer Trends and Technology
ISSN: 2231-2803 / https://doi.org/10.14445/2231280453/IJCTT-V67I3P108
Volume 67 Issue 3, 29-32, March 2019

Original Article

A Novel Spatial Query Search Implementation With Route Server And Location-Based Search

P. Kavya Nikeeta[1,] T. Shronitha[2], T. Nikhila[3], V.V. Sivani [4], P. Dinesh[5]

[1,2,3,4,5] *B.Tech(IV/IV) Dept. of Computer Science and Engineering, Sanketika Vidya Parishad Engineering College, Visakhapatnam, India*

Abstract *- We propose an empirical model of route-based nearest neighbor-based search, which maintains a set of data objects and features at every node end. Initially, the user request can be forwarded to the location-based system and then to the route generator, and it computes the path and return to the requested server and neighborhood nodes as per the received request. In our project model, a path can be generated with our novel approach, and data can be retrieved only from the nodes in the path. If the required number of results is not found in the nearest neighbor, it moves to a further neighbor level. Our cluster-based approach gives more efficient results than traditional approaches.*

Keywords *- nearest neighbor-based search, spatial query search implementation, route generator, location-based search.*

I. INTRODUCTION

Figuring most brief ways effectively in a dynamic chart environment additionally discovers its application in a spatial database framework. In such a framework, it is fundamental to give the usefulness of finding an ideal course in a system. By and large, a chart in a course question framework is of a self-assertive size and is excessively colossal, making it impossible to be the primary memory occupant. In the previous decade, a famous way to deal with taking care of the versatility issue depended on chart apportion. The entire chart is initially apportioned into littler estimated parts, each of which can fit into the primary memory. Since the span of a chart could be subjectively extensive, to accelerate the inquiry procedure and to minimize the I/O movement, a typical strategy is to emerge, in every section, the (nearby) briefest separation data between the alleged fringe vertices (those common by more than one piece).

In a continuous movement data framework, an edge weight in a section could be overhauled powerfully; the briefest separation data between fringe vertices must be re-processed quickly to be valuable in a course inquiry assessment. This can be expert by emerging, for every fringe vertex, a Shortest Path Tree (SPT) to all other fringe hubs in a section; and re-figuring each SPT at whatever point some edge weights in the piece have been changed. Consider an application in which various dispersion bases are scattered in a metropolitan zone. It is valuable to know the minimum cost activity courses from every area to all major crossing points. Taking crossing points as vertices, squares between two convergences as edges, and movement latencies as edge weights, the city activity guide is a digraph with non-negative edge weights.

A spatial database is a database that is optimized to store and query data that represents objects defined in a geometric space. Most spatial databases represent simple geometric objects such as points, lines, and polygons. Some spatial databases handle complex structures such as 3D objects, topological coverages, linear networks, and TINs. While typical databases have developed to manage various numeric and character data types, such databases require additional functionality to process spatial data types efficiently, and developers have often added geometry or feature data types. The Open Geospatial Consortium developed the Simple Features specification (first released in 1997) and sets standards for adding spatial functionality to database systems. The SQL/MM Spatial ISO/EIC standard is a part of the SQL/MM multimedia standard and extends the Simple Features standard with data types that support circular interpolations.

The minimum cost course inquiry between two convergences is to locate a most limited way between two vertices in the comparing diagram. Since the activity condition changes quickly, minimum-cost courses may not be right a couple of minutes after they are figured. More than once, one could apply Dijkstra's calculation to process the briefest ways. Be that as it may, this very much contemplated static calculation may be ineffectual when just a few city streets experience inertness changes. Consequently, specialists have been considering incremental calculations to minimize the briefest re-calculation time.

II. LITERATURE SURVEY

A. NEAREST NEIGHBORS COMPUTATION

"Dynamic Clustering" (DC) is a term specific to the Robocode world that combines a k-nearest neighbor algorithm with kernel density estimation. The term, coined by ABC, refers to k-means clustering but has a lot more in common with k-nearest neighbors. The core idea behind a DC system is that for each decision you make, you examine the current battle situation, compare it to a log of previous situations to find those that are most similar, then use the data collected from those previous situations to decide what to do. This allows your bot to adapt to how much data it has collected so far and change how it classifies that data on-the-fly because it's always re-examining the original data.

B. DYNAMIC CLUSTERS

A dynamic cluster is a server cluster that uses weights and workload management to balance the workloads of its cluster members dynamically, based on performance information collected from the cluster members. Dynamic clusters enable application server virtualization. A dynamic cluster is an application deployment target that can expand and contract depending on the workload in your environment. Dynamic clusters work with autonomic managers, including the application placement controller and the dynamic workload manager, to maximize the use of your computing resources. Dynamic clusters are required for many Intelligent Management autonomic functions, including high availability and service policies.

III. EXISTING SYSTEM

Various authors have proposed various traditional approaches for years of research, and every approach has its advantages and disadvantages. Performance and time complexity are the major factors while community searches. Nodes should be grouped based on the weights and edges existing between the nodes. Traditional community-based approaches are more complex to the group from the source node, and there is no further practical search implementation. Identification of neighbor with simple edge does not retrieve optimality.

A. DISADVANTAGES

- More time complexity if route API computes path with all available nodes for every request.
- Less performance and additional overhead to location-based service.
- Users may receive irrelevant results if the response is slow.

IV. PROPOSED SYSTEM

In an evolutionary approach for efficient Cluster-based route implementation and search implementation, nodes can be grouped with a clustering approach based on weights and edges in terms of graphical nodes and edges format. A node attracts its neighboring individuals to be a part of its path computation. Those that find enough connectivity may choose to stay. The communities then expand further as the newly added members iterate the process. The nearest neighbors can do search implementation, and if it does not get the required limit, it moves to the next search community.

The main advantage here is that the Clustering process reduces the number of nodes. At the same time, computation of paths and irrelevant paths can be ignored, and with less time complexity, the user can receive query results in optimal time.

A. ADVANTAGES

- The clustering process reduces the number of nodes while computation of paths.
- Irrelevant paths can be ignored.
- The user can receive query results in optimal time with less time complexity.

V. ALGORITHMS

A. K-MIDCENTERCLUSTER IMPLEMENTATION

Usually, various nodes available in various locations or zones can be based on the latitude and longitude of the nodes. Nodes can be clustered based on the latitude and longitudes of the nodes. The distance can be computed based on the distance between the centroid and search nodes and gets the minimum distance node, keeps the node in respective clusters, and eliminates unnecessary clusters that are not in zones.

Step1: Load the set of all nodes from various zones and input the search node

Step2:Specify k number of centroids in all nodes (N) and N>=k

Step3: Compute the Euclidean distance between centroid and node N_i

Step4 : While (Euclidean distance(C_i,O_j) <=initial distance) then

Optimal distance:= Euclidean distance;

Centroid_id=Ci;

End while

Step 5: Reinitiate the clusters with new centroids for every iteration

Step6: Continue the process or steps from 2 to 5.

The user requests the location-based service; in turn, it returns the query-based results concerning the features of the object. The user

receives user interesting results based on the features of the requested query. The cache can be maintained if the user makes the same request with minimum time duration and latitude and longitude parameters.

B. LOCATION QUERY SEARCH IMPLEMENTATION:

Input: Qi—Input Spatial Query, DO_{list}(Total Data objects)

Output: R_{list} (result set)

1. The user provides the spatial query, which involves the spatial object and feature.
2. Load DO_{list} from database(LBS)
3. For i=0;i<List_Nodes ;i++

   For each Object O in DO_{list}
   If(O== Q_i.objectname)

   Add to Object_List

   Next for each object O in Object_List

   If (O.attribute==Q_i.attribute)

   Add 'O' to R_{list}

   Next
   Next
4. Sort the Result set
5. Return R_{list}

LBS receives the request from the user, forwards the geocoding's of the user to the route generator and receives the shortest or optimal path, and retrieves the object-based results from the nodes by using a road network; it maintains the objects with features. LBS retrieves the results from the nodes and checks the required threshold. If it does not meet, check for the next immediate neighbor until it meets the threshold value.

VI. RESULT

SCREENS :

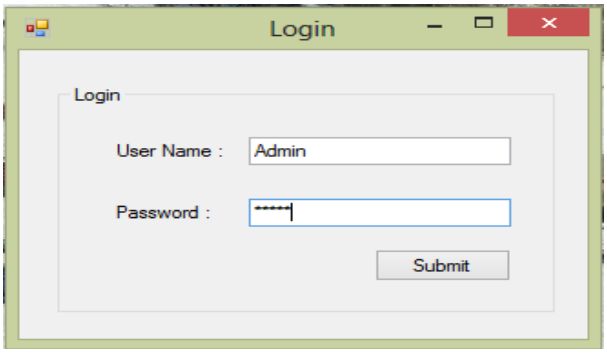

Fig. 1 Login Page

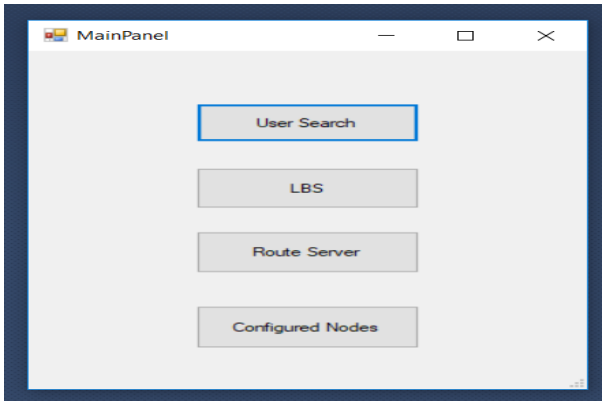

Fig. 2 MainPanel

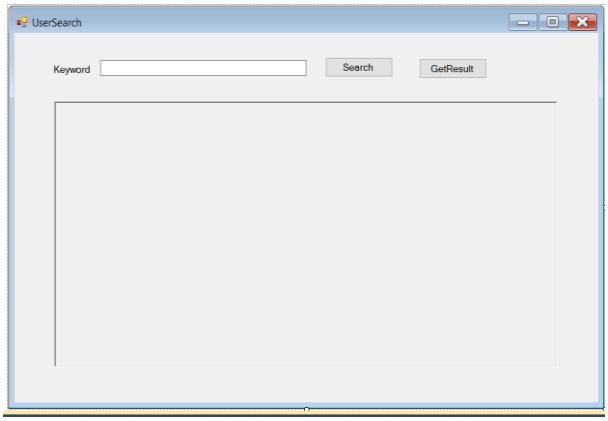

Fig. 3 UserSearch Panel

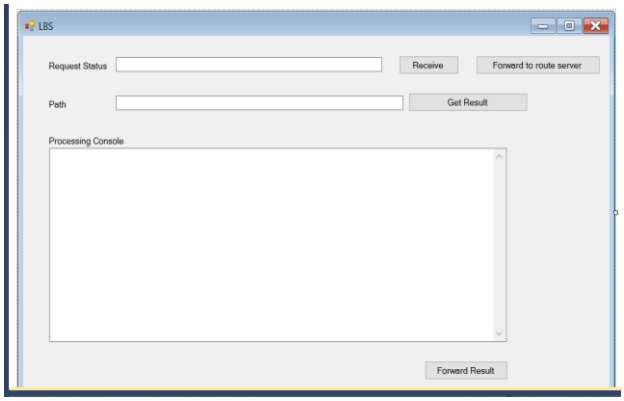

Fig. 4 LBS Panel

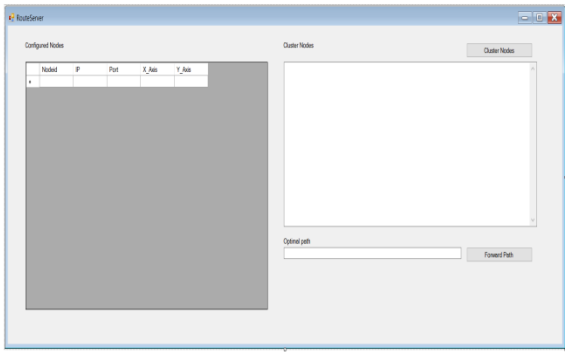

Fig. 5 RouteServer Panel

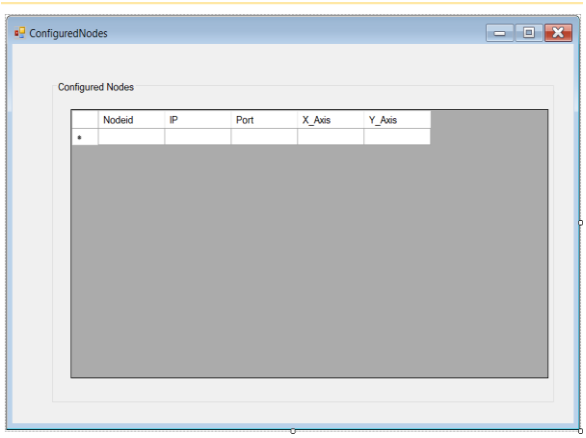

Fig. 6 ConfiguredNodes Panel

VII. CONCLUSION AND FUTURE WORK

We have concluded our current research work with efficient route server-based implementation. User queries can be forwarded to the router server for cluster-based path implementation and then query-based results at the location-based system. Data can be retrieved based on the compared feature of object and attribute of feature, and they are integrated and forwarded. This proposed model gives more efficient results than traditional models.

We can improve the current cluster-based approach with dynamic cluster implementation; in our current approach, we implemented node clustering based on the geo-locations with a static number of clusters, but in the real-time application, data should be grouped or clusters based on data dynamically and if it can support multi-dimensional data then we can improve performance.

REFERENCES

[1] Statistics of Usage. (2013). [Online]. Available: http://www. quantcast.com

[2] US Maps from Government. (2013). [Online]. Available: http:// www.usgs.gov/pubprod/

[3] N. Bruno, S. Chaudhuri, and L. Gravano, "STHoles: A multi-dimensional workload-aware histogram," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2001, pp. 211–222.

[4] E. P. F. Chan and Y. Yang, "Shortest path tree computation in dynamic graphs," IEEE Trans. Comput., vol. 58, no. 4, pp. 541–557, Apr. 2009.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms. Cambridge, MA, USA: MIT Press, 2009.

[6] U. Demiryurek, F. B. Kashani, C. Shahabi, and A. Ranganathan, "Online computation of fastest path intime-dependent spatial networks," in Proc. 12th Int. Symp. Adv. Spatial-Temporal Databases, 2011, pp. 92–111.

[7] Dingle and T. Party, "Web cache coherence," Comput.Netw., vol. 28, pp. 907–920, 1996.

[8] M. Drozdowski, Scheduling for Parallel Processing, 1st ed. New York, NY, USA: Springer, 2009.

[9] H. Hu, D. L. Lee, and V. C. S. Lee, "Distance indexing on road networks," in Proc. 32nd Int. Conf. Very Large Data Bases, 2006, pp. 894–905.

[10] S. Jung and S. Pramanik, "An efficient path computation model for hierarchically structured topographical road maps," IEEE Trans. Knowl. Data Eng., vol. 14, no, 5, pp. 1029–10.

AUTHORS PROFILE

P. Kavya Nikeeta is currently working as an assistant professor in the Computer Science and Engineering Department at Sanketika Vidya Parishad Engineering College.

Thunga Shronitha is pursuing B.Tech from Computer Science and Engineering department at Sanketika Vidya Parishad Engineering College.

Thummalagunti Nikhila is pursuing B.Tech from Computer Science and Engineering department at Sanketika Vidya Parishad Engineering College.

Veerini Venkata Sivani is pursuing B.Tech from Computer Science and Engineering department at Sanketika Vidya Parishad Engineering College.

P. Sai Dinesh is pursuing B.Tech from Computer Science and Technology department at Sanketika Vidya Parishad Engineering College.