EFFICIENT SPATIAL DATA HEURISTIC PARTITION USING VORONOI DIAGRAM OVER DYNAMIC LOCATION DATA THROUGH DECENTRALIZED SERVER

Mr.P.Kumar,P.Dhivya,S.Aslam Gowthar

Assistant Professor, PG Scholar

Department of Computer Science and Engineering
Nandha College of Technology
csekumar@gmail.com, dhivyapalanisamy1993@gmail.com, aslamgowthar786@gmail.com

Abstract

A static index is remodeled sporadically from scratch instead of updated incrementally. It's been shown that throwaway indices be at specialized moving object indices that maintain location updates incrementally. However, throwaway indices suffer from measurability thanks to their single-server style and therefore the solely distributed throwaway index (D-MOVIES), extension of a centralized approach, doesn't scale out because the variety of servers will increase, particularly throughout question process section. we have a tendency to propose a distributed throwaway spatial index structure (D-Toss) that not solely scales bent on multiple servers by victimization Associate in Nursing intelligent partitioning technique however additionally scales up since it totally exploits the multi-core CPUs accessible on every server. D-ToSS apace constructs a Voronoi Diagram that incorporates a flat structure creating it an ideal acceptable data processing. for instance, we have a tendency to through an experiment show a twenty five speed in question process compared to D-MOVIES and this gap gets larger because the variety of servers will increase.

I INTRODUCTION

Large-scale information technology has been evolving separate transaction and analytical systems; data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Generally, any of four types of relationships are sought.

Classes and Clusters: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials. Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.

Associations and Sequential patterns: Data can be mined to identify associations. The beer-diaper example is an example of associative mining. Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes. 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 allow representing simple geometric objects such as points, lines and polygons. Some spatial databases handle more complex structures such as 3D objects, topological cover ages, linear networks, and TINs. While typical databases have developed to manage various numeric and character types of data, such databases require additional functionality to process spatial data types efficiently, and developers have often added geometry or feature data types. A parallel database system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries. Although data may be stored in a distributed fashion, the distribution is governed solely by performance considerations. Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel. Centralized and client– server database systems are not powerful enough to handle such applications. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.

II EXISTING SYSTEM

Moving-object index structures maintain location updates incrementally and process only a fraction of the updates using position approximation techniques. However, all of these approaches suffer from limited scalability due to their single-server design. The Static indexes problem is that most of the throwaway indices are designed for the centralized paradigm that is limited to the capabilities of a single server. On the other hand, devising a distributed throwaway index that scales out to multiple servers is not straightforward due to the following reasons. First, tree based indices cannot directly be implemented in a distributed setting by simply assigning an index node to a server, because traditional top-down search unnecessarily overloads the servers near the tree root. Second, it is essential to create equi-sized partitions; otherwise, the server storing the largest partition will become the bottleneck and slow down the index construction. Third, if the partitioning method does not preserve spatial proximity of the objects, the queries need to be forwarded to all servers to ensure accuracy, which reduces query throughput. Finally, since data objects continuously move, static spatial partitioning methods, where each server is assigned to a zip-code, city or grid cell, will create imbalance across the servers over time.

- High Update workload and major scalability problem.
- Throwaway Indices- Suffer from limited scalability due to single server design.
- R-Tree/ Linearized kd-Tree – Top Down search unnecessarily overloads, reduce query throughput and server access for imbalance.
- D-MOVIES - Query throughput degrades as the number of servers (nodes) increases due to the high query coordination cost.
- RT-CAN is not designed to handle dynamic datasets.
- Spatial Data only used for moving objects.

2.1 Drawbacks

- A distributed throwaway index that scales out to multiple servers is not straightforward
- The objects are broken with hashing.
- Requires a large number of message-passing across the nodes to process updates, Imbalanced grid partition.

III PROPOSED STSTEM

Propose a distributed throwaway temporal spatial index that scales near-linearly both in index construction and Slide Window query processing. The fundamental construct of our index, dubbed (for Distributed Throwaway Spatial Index Structure),is a Voronoi diagram (VD) where we adapt a Voronoi based partitioning method also used Heuristic Partition Algorithm and build a Voronoi cell (VC) for each data object in a distributed fashion. Selected VD a sour partitioning technique and underlying index structure because 1) it has a flat structure that lends itself nicely to parallel processing since each Voronoi cell can be built autonomously in parallel and 2) it is an extremely efficient data structure to answer a wide range of spatial queries. The main challenge in distributed Voronoi Diagram generation is that, due to partitioning, Voronoi cells might be inaccurate because some of their neighboring data objects may reside in a different server so salve this problem to use Heuristic Partition Algorithm. The intuitive approach to overcome this issue is to first build the global Voronoi over all the data objects on a single server and then partition it across the servers for query processing. However, this build- first-distribute-later approach suffers from limited scalability. To overcome this challenge, we propose a novel three- step distribute first-build-later scalable framework. The sliding-window queries that define the authorized view of the data stream for each role.

3.1 Advantages

- The multi-core architecture of each server node.
- All servers communicate with each other in parallel.
- Multi-core parallelization during the local index (local VD) construction. Efficient partition creation in Hueristic Partition.

IV ALGORITHM DESCRIPTION

4.1 Voronoi Diagram Algorithms

A Voronoi diagram decomposes a space into disjoint polygons (cells) based on the set of generators (i.e., data points). Given a set of generators S in the Euclidean space, Voronoi diagram associates all locations in the plane to their closest generator. Each generator s has a Voronoi cell consisting of all points closer to s than other generators.

4.2 DKNN - Distributed K-Nearest Niebuhr Search Algorithm.

A given query q, DKNN can directly identify the set of strips that are guaranteed to contain k neighbors of q, which call the candidate strips.

Step1: Calculating the number of candidate strips.

Step2: Identifying the set of candidate strips.

Finding Three NN using D-KNN

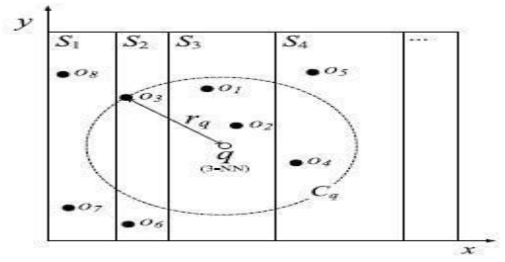


FIG:DKNN Algorithm

4.3 Slide-Window Query

- Query imprecision is defined as the total sum of false-positives and false-negatives for a predicate sliding-window query evaluated on an anonymized Data Stream.
- Slide query is evaluated for all the tuples in equivalence classes that overlap the query region. A utility measure that for a given sliding-window query captures two types of information loss; loss due to generalization (in terms of false positives) and loss due to publishing delay (in terms of false negatives)

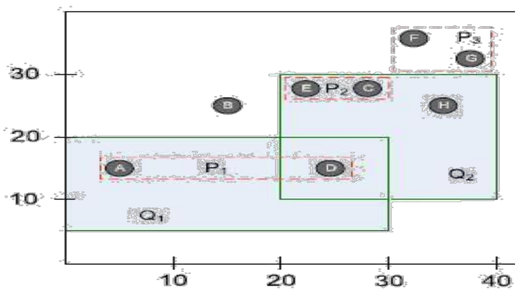


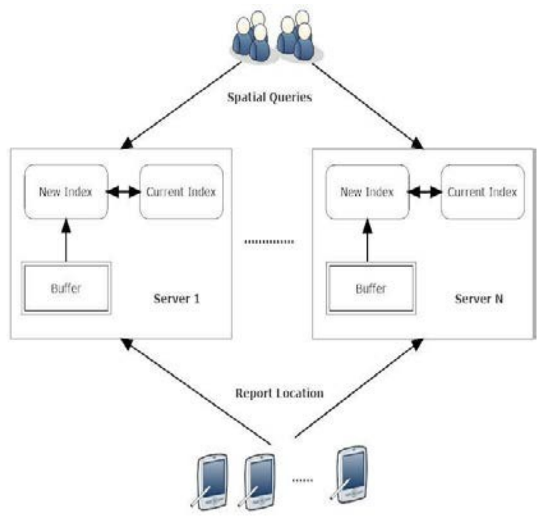
FIG:Slide Window Query Algorithm

4.4 Heuristic Partition Algorithm

```
Algorithm 1: TDH
    Input  : $T$, $k$, $Q$, and $B_{Q_j}$
    Output: $P$
 1  Initialize Set of Candidate Partitions($CP \leftarrow$
 2  for ($CP_i \in CP$) do
 3      Find the set of queries $QO$ that overlap
        such that $ic_{CP_i}^{QO_j} > 0$
 4      Sort queries $QO$ in increasing order of
 5      while (feasible cut is not found) do
 6          Select query from $QO$
 7          Create query cuts in each dimension
 8          Select dimension and cut having leas
            overall imprecision for all queries in
 9      if (Feasible cut found) then
10          Create new partitions and add to $C$
11      else
12          Split $CP_i$ recursively along median t
            anonymity requirement is satisfied
13          Compact new partitions and add to
14  return ($P$)
```

V. SYSTEM ARCHITECTURE

Flow of D-ToSS where each server stores a subset of the data objects and corresponding local index. As shown, while the current locations are collected from the data objects, the queries are processed through the current indices. This separation allows us to run data partitioning phase the background without affecting the performance. Obviously, data partitioning phase does not need to be executed at every index construction cycle since the data distribution does not change significantly within several seconds.



5.1 Voronoi Diagram Construction Module

Generate partial Voronoi diagrams (PVD) in each node in parallel and merge the (partial) results in a single node to build one global Voronoi Diagram. Every Node has a Local Voronoi Diagram for easily lookup the data object in query time

5.2 Pivot Selection and Partition Module

Compute the total sum of the distances between every two objects and choose the set with the maximum total sum as pivots. A partition can add a false-positive tuple to a predicate sliding-window query due to a spatial overlap (QI attributes), temporal (time-stamp attribute) overlap, or both temporal and spatial overlaps.
Voronoi-based adaptive partitioning technique that quickly learns from the dataset and distributes the objects across the servers while preserving the spatial proximity among the objects and balancing load among the servers globally. Second, to take advantage of the multi-core architecture of each server node, generate local Voronoi diagrams (local VDs) using multiple threads where every Voronoi cell is generated by a thread autonomously.

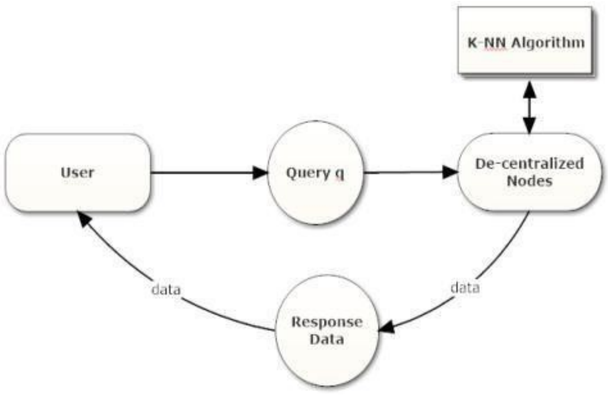
5.3 Data Object Distribution Module

The data object divided into equally sized partition using voronoi cell diagram or Heuristic Partition algorithm. Finally all data sent to all other node in which local voronoi diagram are created independently. All partition data will store local storage and also maintain local index overall index maintain for global node name global index.

5.4 Query Module

D-ToSS to evaluate two major types of spatial queries: range and Distributed k nearest neighbor (Dk-NN).

With D-ToSS, the queries can be submitted to any node in the system. The Dk-NN algorithm works on distributed manner to match nearest data for global partition and find the result forward to user.



VI CONCLUSION

D-ToSS, to index highly dynamic moving object data. The main idea behind D-ToSS is to build short-lived Voronoi based index structures with combination of Slide Window Query instead of updating indexes with each location update from the moving objects. D-ToSS creates the throwaway indexes in a very short time by employing a fully decentralized parallel and distributed architecture that uses multiple server nodes in a cluster. An efficient parallel search algorithm (DKNN) with which queries can be submitted to any node in the cluster without looking for the relevant node that may include the result-sets which provides decentralization unlike tree based approaches. This project successfully Developed for C#.net with SQL server efficiently.

VII REFERENCES

[1]Akdogan, U. Demiryurek, F. Banaei-Kashani, and C. Shahabi. “Voronoi-based geospatial query processing with MapReduce,” in Proc. Cloud Comput. Technol. Sci., Dec. 2010, pp. 9–16.

[2]S. Barahmand and S. Ghandeharizadeh, “BG: A benchmark to evaluate interactive social networking actions,” in Proc. Conf. Innovative Data Syst. Res., Feb. 2013.

[3]Cary, Z. Sun, V. Hristidis, and N. Rishe, “Experiences on processing spatial data with mapreduce,” in Proc. Sci. Stat. Database Manage., 2009, pp. 302–319.

[4]S. Chen, B. C. Ooi, K. L. Tan, and M. A. Nascimento, “A self-tunable spatio-temporal Bþ-tree index for moving objects,” in Proc. Spec. Int. Group Manage. Data, 2008, pp. 29–42.

[5]Y. Tao, D. Papadias, and J. Sun, “The TPR_-tree: An optimizedspatio-temporal access method for predictive queries,” in Proc. Very Large Data Bases, 2003, pp. 790–801.

[6]U. Demiryurek and C. Shahabi, “Indexing network voronoi diagrams,” in Proc. Database Syst. Adv. Appl., 2012, pp. 526–543.

[7]L. Ding, B. Qiao, G. Wang, and Chen Chen, “An efficient quadtree based index structure for cloud data management,” in Proc. Int. Conf. Web-Age Inf. Manage., 2011, pp. 238–250.

[8]J. Dittrich, L. Blunschi, and M. A. Vaz Salles, “MOVIES: Indexing moving objects by shooting index images,” Geoinformatica, vol. 15, no. 4, pp. 727–767, 2011.

[9]R. A. Finkel and J. L. Bentley, “Quad Trees a Data Structure for Retrieval on Composite Keys,” Acta Informatica vol. 4, Mar. 1974, pp. 1–9.

[10]Gold and P. Angel, “Voronoi Hierarchies,” in Geographic Information Science, 2006, pp. 99–111.