Range Tree Based Indexing of Mobile Tracking System

Thu Thu Zan, Sabai Phyu
University of Computer Studies, Yangon
thuthuzan@ucsy.edu.mm, sabaiphyu@ucsy.edu.mm

Abstract

With advances in location-based services, indexing the need for storing and processing continuously moving data arises in a wide variety of applications. Some traditional spatial index structures are not suitable for storing these moving positions because of their unbalance structure. Searching an unbalanced tree may require traversing an arbitrary and unpredictable number of nodes and pointers. Presorting before tree structure is one of the ways of building a balanced two dimensional tree. In this paper, we proposed Presort Range tree that is suitable for moving objects with the dynamic range query. Moreover, with extending mobile technology, tracking the changing position of devices becomes a new challenge. The current location of each user would always be known at the server side whereas it would create a problem. If the mobile movements are small and frequent, at that time unnecessary updates would be performed at the server. In this paper, we also proposed Hybrid Update Algorithm to reduce the server update cost greatly.

Keywords- Location Update Policies, Location Based Service (LBS), Range Tree, Tracking, 2D Range Query

1. Introduction

Everyone who is in IT field says "Today is the age of three things: Cloud Computing, Internet of Things, and Mobile." This word is true because there is no doubt that businesses can reap huge benefits from them [2].

In mobile technology, tracking moving objects are one of the most common requirements for many location management services. Since, the location of moving object changes continuously but the database location of the moving object cannot be updated continuously; therefore, an updating strategy for moving object is required.

In summary, in this paper we introduce the presort range tree for dynamic attributes whose main contributions are as follows.

i. Presort Range tree structure is proposed for moving objects with the availability of dynamic range query.

ii. Hybrid Update Algorithm is proposed that will help to get the current position of moving mobiles at client side and reduces the server update cost greatly.

We explain how to incorporate dynamic attributes in presort range tree and a model is added to deal with overall system. Finally, we made a comparison that will show the experimental result based on presort range tree and without tree. Furthermore, an experimental result of threshold value for proposed Hybrid Update algorithm is made with simulation.

2. The Range Tree

A tree data structure is a powerful tool for organizing data objects based on keys. It is equally useful for organizing multiple mobile objects in terms of hierarchical relationships. There is an assumption for mobile locations that no two points have the same x-coordinate and also y-coordinate. To construct spatial tree structure, the first thing is preprocessing the data into the data structure. Then, queries and updates on the data structure are performed. Then, we treat range query as 2 nested one-dimensional queries: [x1,x2] by [y1,y2]. The first step is to ask for the points with x-coordinates in the given range [x1,x2] => a set of subtrees. Then, instead of all points in these subtrees, only want those that fall in [y1,y2]. In figure, P(u) is the set of points under *u* store *those* points in another tree Y(u), keyed by the y-dimension.

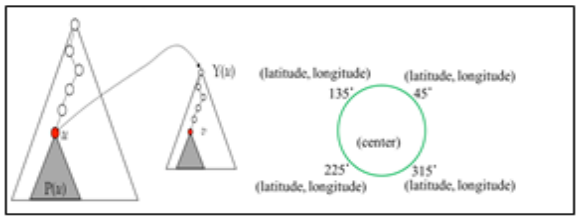

Figure 1: Structure of a Range Tree and Circular Range Searching

3. Location based services

Location based services (LBS) are services offered through a mobile phone and take into account the device's geographical location.

LBS typically provide information or entertainment. LBS largely depend on the mobile user's location. These services can be classified into two types: Pull and Push. In a Pull type, the user has to actively request for information. In a Push type of service, the user receives information from the service provider without requesting it at that instant [4].

3.1. Location Update Policies

Various location update strategies are available in the mobile computing. They are divided into specific strategies like (1) Distance Based Location Update (2) Time Based Location Update (3) Movement Based Location Update (4) Profile Based Location Update and (5) Deviation Based Location Update [1].

3.2. Getting Mobile Location Framework

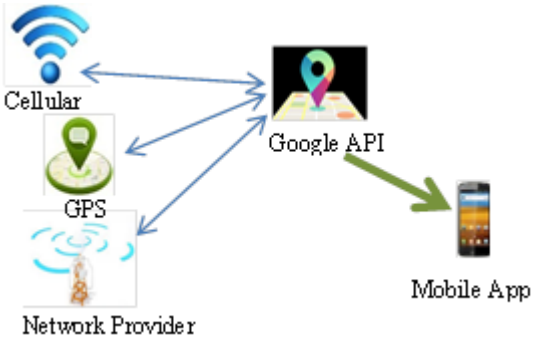

Figure 2: Getting location for Mobile Application

This system has to get current location as fast as it can so it has a framework shown in figure2. It helps to provide a more powerful location framework than usual. This framework is intended to automatically handle location provider's support, accurate location, and update scheduling. It includes the following features.

(a) GPS features –> (GPS, AGPS):.

i. determines location using satellites.
ii. does not need any kind of internet or wireless connection.
iii. depending on conditions, this provider may take a while to return a location fix.

(b) Network provider –> (AGPS, CellID, WiFi MACID):

i.determines location based on availability of cell tower and WiFi access points.
ii.results are retrieved by means of a network lookup.

(c) Cellular Network –> (CellID, WiFi MACID):

i. a special location provider for receiving locations without actually initiating a location fix.
ii. although if the GPS is not enabled this provider might only return coarse fixes.
iii. is mapped to the specific set of hardware and telecom provided capabilities

To shorten the time to first fix, or the initial positioning or increase the precision in situations when there is a low satellite visibility, the mobile network should be used. The best way is to use the "network" or "Cellular Network" provider first, and then fallback on "gps", and depending on the task, switch between providers.

4. Related Works

There are a number of papers that describe about moving objects' index tree structure and mobile update policies. Most papers are focus on using one index structure and one update policy. Some discuss combination of index trees called hybrid tree structure and comparison of using one index structure and it.

Dongseop Kwon, Sangjun Lee, Sukho Lee proposed a novel R-tree based indexing technique called LUR-tree. This technique updates the structure of the index only when an object moves out of the corresponding MBR (minimum bounding rectangle). If a new position of an object is in the MBR, it changes only the position of the object in the leaf node. [5]**.** So, it remove unnecessary modification of the tree while updating the positions because this technique updates the index structure only when an object moves out of the corresponding MBR (minimum bounding rectangle.

Christian S. Jensen, Dan Lin, Beng Chin Ooi represented moving-object locations as vectors that are time stamped based on their update time. By applying a novel linearization technique to these values, it is possible to index the resulting values using a single B^+tree that partitions values according to their timestamp and otherwise preserves spatial proximity. This scheme uses a new linearization technique that exploits the volatility of the data values, i.e., moving-object locations, being indexed. Algorithms are provided for range and _NN queries on the current or near-future positions of the indexed objects [4].

Yuni Xia, Sunil Prabhakar proposed a novel indexing structure, namely the Q+Rtree that is a hybrid tree structure which consists of both an R*tree and a QuadTree. In Q+R tree, quasi-static objects are stored in

an R*tree and fast-moving objects are stored in a Quadtree. By handling different types of moving objects separately, this index structure more accurately reflects the reality and results in better performance. In their work, no assumption is made about the future positions of objects. It is not necessary for objects to move according to well-behaved patterns and there are no restrictions, like the maximum velocity, placed on objects either [7].

Cheng, Pingzhi Fan, Xianfu Lei, and Rose Qingyang Hu made a location update scheme in which update occurs either when the movement threshold for MBLUs is reached or when the time threshold for TBLUs is reached. The movement counters and the periodic LU timer reset when an LU occurs. They used convex function of the movement threshold. That is, there is a value of the movement threshold that can minimize the signaling cost. They showed that the HMTBLU scheme always has higher signaling cost than the MBLU scheme [3].

Vicente Casares_Giner, Pablo Garcia-Escalle proposed a location update scheme by combining two dynamic strategies, movement based and the distance based [6]. They showed that results obtained from these analytical model show that, with little memory requirements in the mobile terminal very good performances can be obtained. However, it required that after each movement, the mobile terminal has to search the identity of the new visited cell in a cache memory.

5. Proposed Approach

This paper is integrated by two major components: client side and server side. The overall system model is built and hybrid update algorithm that will aid to get last current location and reduce server update cost is proposed at the client side. Presort range tree procedure for moving objects is included in the server side.

5.1. System Model

A model, searchable model is built to incorporate dynamic attributes in presort range tree and query processing. This includes a server and a collection of registered mobile objects. In order to keep the location information up to date, these objects regularly send their updated positions to the server. Unnecessary updates wouldn't be performed at the server because Hybrid Update Algorithm is applied to the client side. The require information query the server with range queries like "which mobiles are currently located within a disaster area?" To process such queries efficiently, the server maintains an index tree that, in addition to speeding up the query processing, is also able to absorb all of the incoming updates.

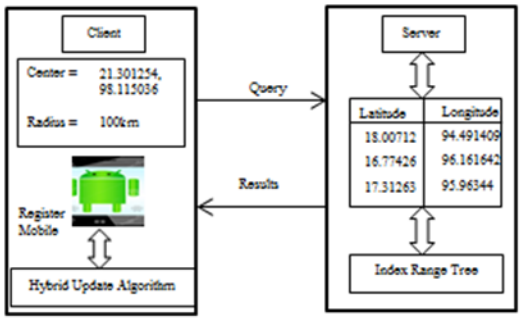

Figure 3: Client-Server System Model

5.2. Hybrid Update Algorithm

The distance-based update scheme seems to be simple location update strategy. In this scheme, each mobile host has to track the distance it moved since its last location update. When the distance exceeds the threshold (HD), the mobile host transmits an update message.

But it is complicated because of the variation of cell sizes and the need to compute the distance a mobile has moved.

The time based location update strategy is a simple strategy for location update. Here the mobile base station would update the location of user after a particular time period say T. However, the main drawback here would be sometimes if the user is stationary at that time unnecessary updates would be performed.

In this paper, hybrid location update algorithm is proposed based on time and distance so that it can significantly reduce location update overhead which improves the efficiency of mobility support mechanisms. The structure of mobile location update is shown in figure.

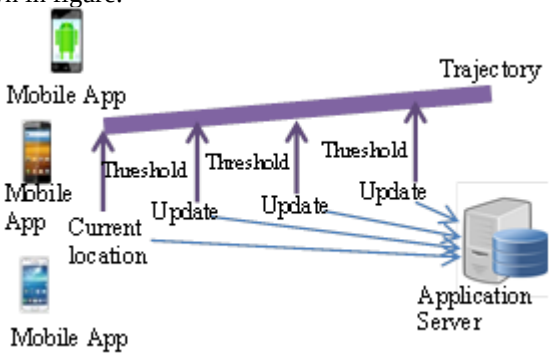

Figure 4: Mobile Location Update Structure

The advantage of the proposed algorithm is that it reduces location update traffic, with a minimum increase in implementation complexity.

Algorithm: Hybrid Location Update
Input: Database of mobile locations contains the locations of registered mobile with time
Output: current registered mobile location

1. i=0; dis_threshold; time_threshold; time_scheduler;

2. Read the current location (L_{xi},L_{yi},t_i) and previous location(L_{xi-1},L_{yi-1},t_{i-1}) of registered mobile location

3. If the time_scheduler > time_threshold &&

$\sqrt{(L_{xi}-L_{xi}-1)^2 +(L_{yi}-L_{yi}-1)^2} > dis\_threshold$

4. 3.1. Update the database with current location (L_{xi-1}, Lyi_{-1},t_{i-1}) = current location (L_{xi},L_{yi},t_i),i+1.

   3.2. Total number of update=Total number of update+1;
5. Else current location(L_{xi},L_{yi},ti) = previous location (L_{xi-1},L_{yi-1},t_{i-1}), i=i+1;

5.3. Proposed Presort Range Tree

The procedure of proposed presort range tree is the following;
Input: Lats=Array of two dimensional points sort on latitudes
Longs=Array of two dimensional points sort on longitudes
Procedure PRTree (Lats, Longs)

1. If Lats.length==1 then return new LeafNode(Lats[1]);
2. medium= [Lats.length/2];
3. Copy Lats[1….medium] to $Lats_L$ and Lats[medium+1….. Lats.length] to $Lats_R$;
4. for i=1 to Longs.length do
5. if Longs[i].x <= Lats[medium].x then append Longs[i] to $Longs_L$;
6. else append Longs[i] to $Longs_R$;
7. root= new Node((Lats[medium].x),One D Range(Y));
8. root.left= PRTree($Lats_L$, $Longs_L$);
9. root.right= PRTree($Lats_R$, $Longs_R$);
10. return root;

5.3.1. Circular Range Search

After preprocessing of tree construction is done, the structure allows searching circular range query for mobile objects. To determines whether registered mobiles are in service area or not so that this system has to **get bounding coordinates** with center and service distance: (centerLat, centerLong, bearing, distance).

```
bearingRadians = Radians(bearing);
lonRads  = Radians(centerLong);
latRads  = Radians(centerLat);
maxLatRads = asin((sin(latRads) * cos(distance / 6371)
+ cos(latRads)
sin(distance / 6371) * cos(bearingRadians)));
maxLonRads = lonRads + atan2((sin(bearingRadians) *
sin(distance / 6371) cos(latRads)),(cos(distance / 6371) -
sin(latRads) *      sin(maxLatRads)));
```

5.3.2. Example: Calculating Presort Range Tree with center and service distance

Firstly, sort the mobile locations by latitudes and longitudes.

Sort by X	Sort by Y
16.35099 96.44281	16.77923 96.03917
16.77923 96.03917	16.80958 96.12909
16.80958 96.12909	26.69478 96.2094
24.77906 96.3732	24.77906 96.3732
24.99183 96.53019	16.35099 96.44281
25.38048 97.87883	24.99183 96.53019
25.40319 98.11739	26.35797 96.71655
25.59866 98.37863	25.82991 97.72671
25.82991 97.72671	25.38048 97.87883
25.88635 98.12976	25.40319 98.11739
26.15312 98.27074	25.88635 98.12976
26.35797 96.71655	26.15312 98.27074
26.69478 96.2094	25.59866 98.37863

Then the Presort Range Tree is built and shows as the following;

```
25.40319 98.11739
LEFT:     16.80958 96.12909
LEFT:     16.35099 96.44281
RIGHT:    16.77923 96.03917
RIGHT:    24.99183 96.53019
LEFT:     24.77906 96.3732
RIGHT:    25.38048 97.87883
RIGHT:    25.88635 98.12976
LEFT:     25.59866 98.37863
```

RIGHT: 25.82991 97.72671
RIGHT: 26.35797 96.71655
LEFT: 26.15312 98.27074
RIGHT: 26.69478 96.2094

The results of sample range search in centerLat, centerLng, distance: 26.693, 96.208, 1000km that are registered mobile locations to send notification as follows:

node (25.40319, 98.11739)
RIGHT: node (24.99183, 96.53019)
LEFT: node (24.77906, 96.3732)
RIGHT: node (25.38048, 97.87883)
RIGHT: node (25.88635, 98.12976)
LEFT: node (25.59866, 98.37863)
RIGHT: node (25.82991, 97.72671)
RIGHT: node (26.35797, 96.71655)
LEFT: node (26.15312, 98.27074)
RIGHT: node (26.69478, 96.2094)

6. Simulation Results

The simulation considers an experimental result with threshold value for proposed Hybrid Update algorithm. These values inserted in the local database of the moving object. Then compute the distance and if the distance >= a specific threshold, an update occur.
In figure 6 that represent the actual and expected path through 9 minute at threshold = 2 miles. This result shows that central database needs to be update with actual location only five times at point a, b, c, d and e. Since the distance is greater than the value of threshold instead of updating the database every time.

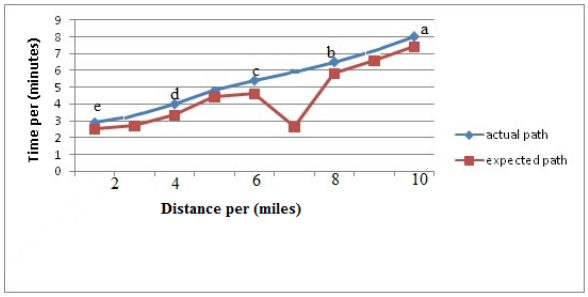

Figure 5.The actual and expected path at (threshold=2 miles)

The next experiment has been performed on a 2.60 GHz ASUS PC, with Intel (R) Core (TM) i7 CPU and 4 GB memory. For this experiment, we use the most popular testing framework in Java, JUnit. It is an open source testing framework which is used to write and run repeatable automated tests. The experiment was performed for computing query and execution time for range search, with number of data set points that are organized in two dimensions. To construct spatial tree structure, the first thing is preprocessing the data into the data structure. Then, queries and updates on the data structure are performed. It has been used to compare the performance of both, Range tree and without Tree, for data set points in a 2-dimentional space. The execution time required by both approaches was different. Query found by both approaches with the same points. Better performance was achieved when the Range tree was used for larger number of data sets for range search. The more volumes of data tests, the less number of seconds needs in Range tree.

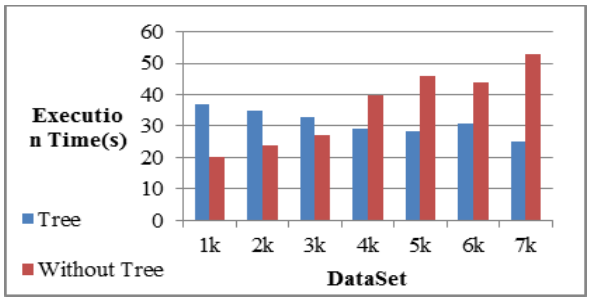

Figure 6. Execution Time (Preprocessing time +Query time) of Tree and Without Tree

7. Conclusion and Future Works

In this paper, the main service task is handling mobile objects based on index tree structure. The system maintains the moving mobile locations and circular range query is available from the server. Therefore, the system is done for monitoring of mobile objects, to be able to efficiently locate and answer queries related to the position of these objects in desire time. The system will helps to be tradeoff frequency of update due to the locations of mobile objects and reduce server update cost. It also support range query with dynamic object locations.
For future works, the proposed Hybrid Update approach will be applied to other index structures (e.g. the quad tree, the K-D-B tree). Moreover, this proposed system can be used to storing other moving objects such as temperature, vehicle location and so on. The results obtained from the other index tree structure can be compared to this paper's results.

8. References

[1] A.Kalpesh A, S.Priyanka, "Various Location Update Strategies in Mobile Computing", International Journal of Computer Applications® (IJCA) (0975 – 8887) Proceedings on National Conference on Emerging Trends in Information & Communication Technology (NCETICT 2013)

[2] Rundle, M.Huffington, "future of technology whitepaper", UK, 2015.

[3] Cheng, Pingzhi Fan, Xianfu Lei, And Rose Qingyang Hu," Cost Analysis Of A Hybrid-Movement-Based And Time-Based Location Update Scheme In Cellular Networks", IEEE Transactions On Vehicular Technology, Vol. 64, No. 11, November 2015.

[4] Christian S. Jensen,_ Dan Lin, Beng Chin Ooi, "Query and Update Efficient B+-Tree Based Indexing of Moving Objects", VLDB 04 Proceedings of the Thirtieth international conference on Very large databases, Volume 30 pages 768-779.

[5] Dongseop Kwon, Sangjun Lee Sukho Lee, "Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree" Third International Conference on Mobile Data Management, IEEE, 2002.

[6] Vicente Casares_Giner, Pablo Garcia-Escalle, "An Hybrid Movement-Distance-Based Location Update strategy for Mobility Tracking", CICYT (Spain) for financial support under project number TIC2001-0956-C04-04.

[7] Yuni Xia Sunil Prabhakar, "Q+Rtree: Efficient Indexing for Moving Object Databases", Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03), March 26-28, 2003, Kyoto, Japan.