

Towards a Unified Index Scheme for Mobile Data and Customer Profiles in a Location-Based Service Environment

Vijayalakshmi Atluri, Nabil R. Adam and Mahmoud Youssef

MSIS Department and Center for Information Management, Integration and Connectivity (CIMIC)
Rutgers University
{atluri,adam,youssefm}@cimic.rutgers.edu

Extended Abstract

1. Introduction

The increase in the demand for applications dealing with moving objects can be seen in recent years. Leaders in the mobile phone industry expect more than 1 billion mobile devices by 2004. Furthermore, mobile phones and/or wireless PDAs are expected to evolve into wireless terminals that are GPS enabled. In addition to wireless computing devices, tracking of other moving objects such as boats, trucks, automobiles, airplanes, and soldiers is also of growing interest. With the great demand on mobile devices, mobile commerce became a gigantic market opportunity. Some researchers predict the global subscriber base for mobile location services to exceed 680 million users by the end of 2006. Among those, 50% are mobile subscribers. They will represent more than 70% of the mobile Internet users. The revenues from mobile services have reached \$2 billion by the end of 2002 and expected to reach \$18 billion by the end of 2006. These revenues are divided as follows: 31% in Western Europe, 22% in the United States, and 47% in Japan and the rest of the World. As such, the market for location-aware mobile applications such as mobile shopping, mobile advertising, mobile retailing and mobile online banking is very promising. A study by Durlacher [2] shows that mobile advertising will be the killer application with 23% of the market share. The needs of such applications go beyond tracking users' locations, for example, they may additionally need to track user profiles and preferences in order to achieve mass personalization. This is because, to be effective, targeted advertising should not overwhelm the mobile consumers and must push information only to a certain segment of mobile consumers based on their preferences and profiles, and based on certain marketing criteria. Obviously, these consumers should be targeted only if they are in the location where the advertisement is applicable at the time of the offer. It is important to note here that user profile information may include both sensitive and non-sensitive attributes such as *name, address, linguistic preference, age group, income level, marital status, education level, etc.*

While mobile consumers like to benefit from personalization, they usually are not willing to share their sensitive profile information to all the merchants. To ensure the privacy of mobile users, it is important that the sensitive profile information is revealed to the respective merchants only on the need-to-know basis. Therefore, it is essential that the profile information be maintained by a third party service, rather than by the merchant's system, to ensure the privacy of the mobile users. Typically, the tracking of mobile objects (consumers), i.e., maintaining the moving object database and responding to queries, is performed by the location service (LS). Obviously, this third party service can be carried out by the LS. This is because it is prudent and economical to use the same service to maintain the profiles, instead of using another service just for this purpose.

Given that databases use indexes for efficient retrieval of data, we need a *unified* index for location data and profile data. We believe that such index would provide a significant gain in performance. This is because, the best query plan would take three steps (1) a multidimensional query on profiles database (let the response time be t_1), spatio-temporal query on moving objects database (let the response time be t_2), and an intersection operation (let the response time be t_3). In contrast, with the uniform index, the query is processed in one pass, and therefore, it is expected that the response time is far less than $t_1 + t_2 + t_3$.

In order to build a uniform index for moving object data and user profile attributes, one has to resort to a multi-dimensional index structure. While some of the solutions are not applicable to multi-dimensional data (e.g., PMR-Quadtrees), others do not scale well (e.g., TPR-trees [3]), as has been shown by Berchtold et al. [1] that the performance of tree-based indexes deteriorates greatly due to the curse of dimensionality. An alternative to the tree structure is the hashing-based index proposed by Song and Roussopoulos [4], where an object's spatial location within certain area is mapped to a bucket in the hash index. They proposed several functions that can be applied to moving objects in high dimensional space, which represent hyper-rectangles in the geometric space. A distance function has been used as a typical hash function, by choosing a random point and computing the d-dimensional distance from it to all other points. Use of a hash function essentially transforms large number of dimensions into a very small number of dimensions, thereby allowing the hashed data to be stored either in a B+-Tree (in case of a single dimension), in a Quadtree or in an R-tree (in case of two or three dimensions). However, the main drawback of this approach is that it results in a large number of false positives. In other words, since a query is transformed due to the hash function, a query results in more objects than that really satisfy it. Given that the two extreme approaches, the tree-based and hashing, the challenge is to build an indexing technique that is efficient as well as produce less false positives.

2. Proposed Approach

In this paper, we propose a novel index structure to manage both profiles and moving object data with the goal to maximize both performance and accuracy of query processing. Queries may range from point-in-time query (e.g., retrieve all female customers who are currently in the shopping mall with age between 18 and 23), time interval queries (e.g., retrieve customers who will pass by the motel with in the next 60 minutes that do not live in Pennsylvania), and continuous query (e.g., retrieve all the customers who are within 300 feet of the store, whose salary is above 20K). Our approach essentially is to cluster the customers based on their profiles using a categorical clustering algorithm, and then construct a TPR tree for each cluster, as shown in Figure 1. Due to the fact that profile data is relatively static, much of the attribute values are either binary or categorical, and there exist considerable correlations among the profile attributes, profile data lends itself to clustering. Our results in Figure 2 indicate that clustering significantly improves performance, and it outperforms a single tree without clustering, provided the query processing does not require visiting 5 or more clusters. To improve the accuracy of the query processing, we perform some preprocessing on the clustered data, which involves the following steps:

Step 1: The preprocessing first sorts the attributes based on their accuracy of clustering, which we measure with a *classification factor* computed as follows: For each attribute in the profile database, we first construct a pivot table. Each cell of this pivot table is comprised of the number of customers in each category in each cluster. For each column in the pivot table, we calculate the sum of the cells that have significant value when compared to the others, and the sum of all the cells in the table (T). We then compute the Classification Factor (A/T), which is then used to sort the attributes in the profile database. We then turn the values in the cells into probabilities by dividing the number of customers in each cell by the total number of customers in the table.

Step 2: We then prune the categories of each attribute. The fact that some attributes have categories that have very little contribution to the classification scheme, removing these categories would help in improving the query performance and in reducing the storage requirement. To accomplish this, we set a pruning threshold (h) and prune a category from an attribute if $combined(x) * m \leq h$, where $combined(x)$ is the combined probability of category x , and m the number of categories.

Step 3: To improve the accuracy further, we attempt to reduce the number of misclassified points. To accomplish this, we classify each point using the following classification approach. For each point, we select the cluster with the maximum probability by examining its probabilities in each attribute, starting with the first attribute from the sorted list of attributes, and classify it to that cluster. We then recluster the data using the pruned classification scheme, re-build the pivot tables, and re-compute the probabilities to eliminate mis-clustered points.

A query is processed by first breaking it into two parts: profile part, and location part. Processing the profile part would result in the target cluster(s). The location part is then processed by traversing this cluster. In case of a point query, for each attribute in the query, we construct an array of probabilities by adding the appropriate column from the pivot table and adding all these columns for each attribute. Then

the query is directed to the cluster(s) that has the maximum probability in the array. In case of a range query, we simply add all the columns representing the categories in the range to the array. We also allow users to specify their desired accuracy level, and achieve this by selecting more clusters to process the query. However, a higher accuracy may result in reduced performance.

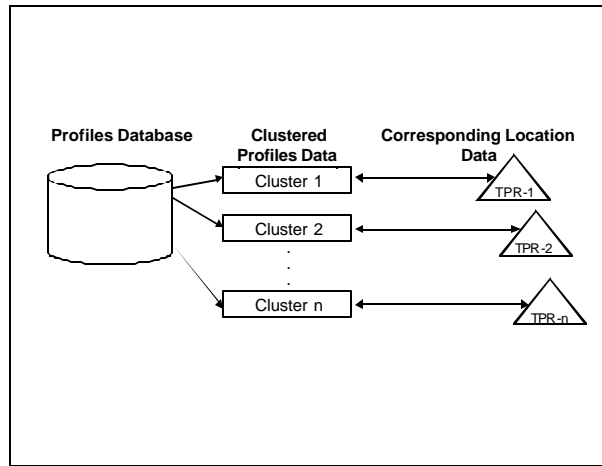


Figure 1: Our Approach

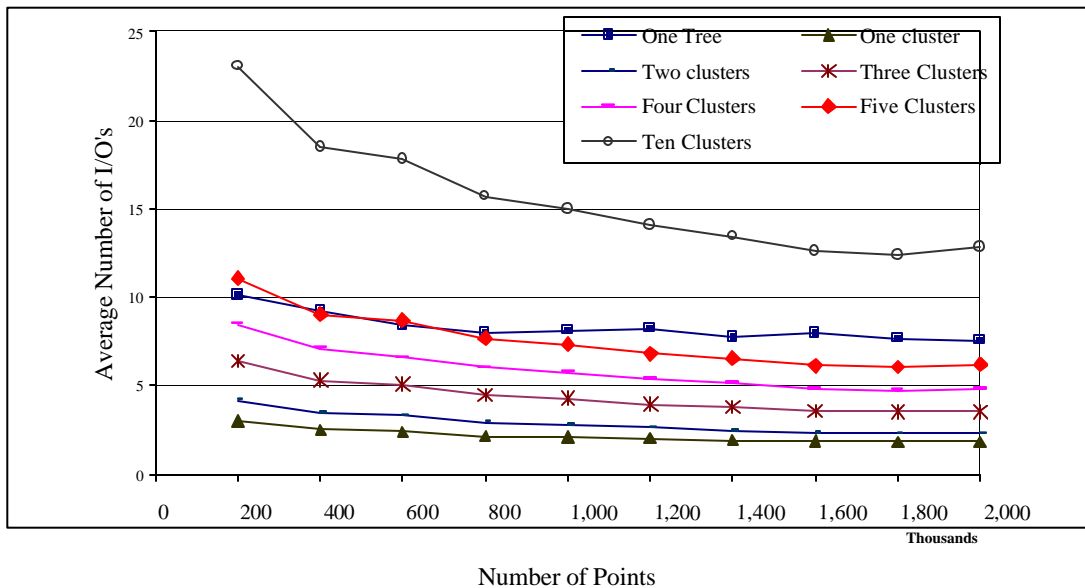


Figure 2: The effect of breaking TPR-tree to multiple trees on the performance

References:

[1] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. Improving the query performance of highdimensional index structures by bulk-load operations. *EDBT'98*.
 [2] Durlacher Research Ltd. *Mobile Commerce Report*. <http://www.duralcher.com>. 2001. *Database Technology, Valencia, Spain, March 23-27, 1998*, volume 1377 of *LNCS*, pages 216–230. Springer, 1998.
 [3] J.S. Saltenis, C. S. Jensen, S. Leutenegger and M. Lopez. *Indexing the Positions of Continuously Moving Objects*. In Proceedings of the ACM-SIGMOD 2000.
 [4] Zhexiong Song and Nick Roussopoulos. Hashing Moving Objects. *Mobile Data Management*, pages 161–172, 2001.