Efficient Processing of K-NN Queries Using Data Indexing Technique

[1]Dipali Ahire, [2] Nikita Kadam, [3] Prajakta Patil, [4] Sujata Thete, [5] Dr.Kalpana Metre

[1]B.E. Student, Information technology, Maharashtra, India
[2] B.E. Student, Information technology, Maharashtra, India
[3] B.E. Student, Information technology, Maharashtra, India
[4] B.E. Student, Information technology, Maharashtra, India

ABSTRACT

Developing an efficient processing techniques in spatio-temporal databases has been nowadays a much discussed topic. Applications, such as mobile information systems, traffic control system, and geographical information systems, can benefit from efficient processing of K-NN queries. Due to the rapid advancements in positioning technologies such as the Global Positioning System (GPS) and wireless communications, it has become convenient for tracking of continuously moving objects. However, this development poses new challenges to database technology since maintaining up-to-date information regarding the location of moving objects incurs an enormous amount of updates. Many applications involving moving objects consists the task of processing k-nearest neighbour (k-NN) queries. Many approaches are designed to this problem, for the centralized setting where query processing is performed on a single server.

Keywords: *Spatio-temporal databases; spatio-temporal queries; K-nearest neighbour queries.*

1. INTRODUCTION

In many applications, to know moving objects current location in advance is very appreciable. Discovery of patterns of future movement can greatly effect on different fields. All moving things in the real world can contain Spatio temporal data, containing time and space attributes simultaneously. K-nearest neighbor (k-NN) queries over moving objects using dynamic strip index is a fundamental operation in many location-based applications. The advances of GPS technology and wide-ranging usage of wireless communication devices have facilitated the collection of large amount of spatiotemporal data. A suite of solutions that can support scalable distributed processing of k-NN queries is proposed. A new index structure called Dynamic Strip Index (DSI) is presented, which can better adapt to different data distributions than exiting grid indexes.

The K-nearest neighbor (KNN) query is an important type of the spatio-temproal queries. Given a set of objects So, a query object q, and a value of K, the KNN query finds the K-nearest neighbors of *q* among So. The problem of processing k-NN queries over moving objects is fundamental in many applications. DSI, a distributed strip index, and DKNN, a distributed k-NN search algorithm, to address this challenge is proposed. Both DSI and DKNN are designed with distributed processing, and can be easily deployed to a distributed system. DSI is a data partitioning index and is able to adapt to different data distributions. Based on DSI, the DKNN algorithm is been presented that can directly determine a region that contains the k -NN for a given query with only two iterations.

When objects move with reasonable velocities, its result (a list of objects) remains relatively stable, for a given k-NN query q. Therefore, it is promising to investigate how the k-NN results can be incrementally updated as objects move. Based on DSI, we present the DKNN algorithm that can directly determine a region that is guaranteed to contain the k-NN for a given query with only two iterations. K-NN can be computed in two ways:

1. By dynamically using an updated grid-based index structure to index objects or queries.

2. Generalization of object indices to hierarchical structures.

The query performance and robustness for skewed object distributions is improved by all this. Query processing is extended to increment and maintain the query answers. When the object's movement is localized, i.e. the velocity is in general bounded; its performance is further improved.

2. LITERATURE SURVEY

The global pool of generated personal location data was least 1PB in 2009 and now it is growing by about 20 percent a year, that is been estimated by our recent studies. In addition to managing the vast volume of data, a variety of location-based applications and services, must be able to effectively handle a large quantity of user-initiated concurrent queries in which many of them are K-NN queries. Solutions that can support the processing of many concurrent k-NN queries over large volumes of moving objects data must be resolved. [1]An increasingly mature model of enterprise IT infrastructure that is Scalable Searching provides from a shared pool of configuration cloud resources, high quality applications and services. The data mining can outsource their local complex data system through individuals or enterprises into the data to avoid the costs maintaining a private storage as possesses powerful functionality and flexibility.

Querying scheme supporting both user and user to access and update the data and dynamic update over data. For more accurate result, contribution is made mainly in two aspects: prediction and showing the storage of data securely and distribute data securely to the user. To evaluate the performance of DKNN, we implement three other algorithms on S4 as baseline methods. The task of deploying DSI on S4 involves two types of PEs, EntrancePE and IndexPE.

3. PROPOSED SYSTEM

Many applications involving moving objects is the task of processing k-nearest neighbor (k-NN) queries. Many approaches to this problem are designed for the centralized setting where query processing takes place on a single server. It is difficult to scale to a distributed setting, that are increasingly common in those applications, to handle the vast volume of data and concurrent queries. DSI and DKNN are implemented on Apache S4, an open-source platform for distributed stream processing. To study the characteristics, an extensive experiment of DSI and DKNN are performed, and then it is compared with three baseline methods. Our proposal scales works well and significantly outperforms the alternative methods.

To allow the users to locate their nearest fellow users upon request is one of its main functionality. In the new era of big data, it is imperative to find solutions that can effectively support the processing of many concurrent k-NN queries over large volumes of moving objects data. The DSI structure and the DKNN algorithm strike a good balance between the cost of index maintenance and query processing. DKNN supports more efficient updates as it has a less complex index structure than the tree-based approaches.

The most notable advantage of DKNN is that even though the master node does not store the positions of objects, it can still determine the search space that contains the k-NNs in just two steps, by first directly determining the candidate strips using the DCS algorithm, and then identifying the final set of strips to search by computing the circle. With those algorithms, the master cannot determine the final region for k-NN search without involving an uncertain number of rounds of communication between the master and slaves, incurring significant communication costs.

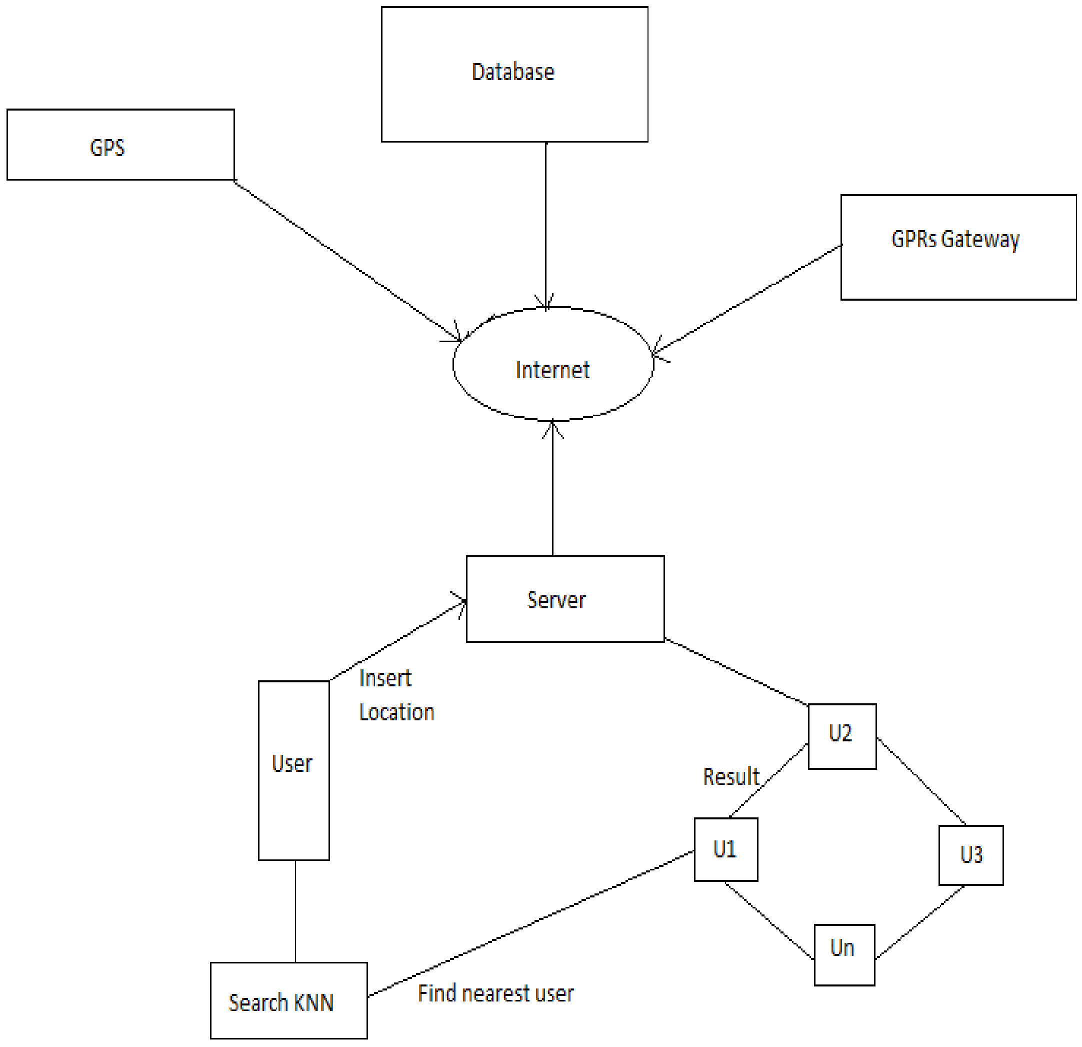

Fig -1: System Architecture

4. Algorithm

Algorithm 1: DKNN Algorithm

Input: The query q (qx; qy); vertical strips LV; horizontal strips LH; the parameter x

Output: The K-nearest neighbors of q.

1: Let the number of vertical and horizontal candidate strips c;

```
2: Let CV =DCS (q(x); q(y), LV, c);
3: Then Compute CH;
4: Set CT = CV [CH];
5: if c (j), CT (j) _2c then
6: Find x supporting objects in every candidate strip, and put   them into c;
7: Then Compute the distances from the supporting objects in to q;
8: Let o be the k-th nearest neighbor of q (9Si 2 CT; s.t. o 2Gi)
9: Set rq = distance (o, q);
10: Determine circle CQ that is centred at q with radius rq;
11: Let F ¼fSjS 2 LV [LH, S intersects CQ];
12: Find the k-NNs from the objects covered by strips in F;
13: end if
14: if 0 < jCT j < c then,
15: Search all strips in LV or LH to obtain k-NNs;
16: end if
17: Return k-NNs;
```

Algorithm 2: DCS Algorithm

Input: The query q [q(x); q(y)]; LV; c.

Output: Vertical candidate strips CV.

```
1: Let CV ¼ f,
2: Sort LV according to the low boundaries of strips;
3: if strip Si satisfies lbi _ qx < ubi then
4: Insert Si into the set CV;
5: CV = find CS (i _ 1, i þ 1);
6: end if
7: if there exist two strips Si, Siþ1, which satisfy Si: ub < qx< Siþ1: lb then
8: CV = find CS (i _ 1, i þ 1);
9: else
10: if 8Si (0 _ i _ n), Si: lb > qx then
```

```
11: Insert the first c elements of LV into CV;
12: else
13: Insert the last c elements of LV into CV;
14: end if
15: return CV;
```

5. CONCLUSIONS

A technique that can quickly identify the related queries of a given query q is introduced in this paper. Our algorithm uses the information to infer a search region that covers the k-NN results of query q. Our algorithm can only deal with k-NN queries. The problem of processing k-NN query over moving objects is fundamental in many applications. New scalable solutions are called by the large volume of data and heavy query workloads. To address this challenge, the DSI (a distributed strip index), and DKNN (a distributed k-NN search algorithm) is used. Both DSI and DKNN can easily be deployed to a distributed system, that are designed with distributed processing.

6. REFERENCES

1. X. Yu, K. Pu, and N. Koudas, "Monitoring k-nearest neighbor queries over moving objects," in Proc. 21st Int. Conf. Data Eng., 2005, pp. 631–642.

2. W. Lu, Y. Shen, S. Chen, and B. C. Ooi, "Efficient processing of k nearest neighbor joins using MapReduce," Proc. VLDB Endowment, vol. 5, no. 10, pp. 1016–1027, 2012.

3. X. Xiong, M. F. Mokbel, and W. G. Aref, "SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatiotemporal databases," in Proc. Int. Conf. Data Eng., 2005, pp. 643–654.

4, W. Wu, W. Guo, and K. Tan, "Distributed processing of moving k-nearest-neighbor query on moving objects," in Proc. Int. Conf. Data Eng., 2007, pp. 1116–1125.