IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

# Main Memory Sharing for the Processed Data using Spitfire AKNN Algorithm

**Ms Rajalakshmi A B[1], Dr. K. S. Shreedhara[2]**

M.Tech, Computer Science and Engineering, University BDT College of Engineering, Davanagere, Karnataka, India[1]

Professor, Department of Computer Science and Engineering, University BDT College of Engineering, Davanagere, Karnataka, India[2]

**Abstract**: A huge spectrum of Internet-scale mobile applications, for example social networking, gaming and entertainment, emergency response and crisis management, all require efficient and scalable All k Nearest Neighbours (AkNN) computations over millions of moving objects every few seconds to be operational. Most traditional techniques for computing AkNN queries are centralized, lacking both scalability and efficiency. Only recently, distributed techniques have been proposed to achieve scalability for large datasets. These batch-oriented algorithms are sub-optimal due to inefficient data space partitioning and data replication among processing units. Algorithm Spitfire is a distributed algorithm that provides a scalable and high performance AkNN processing framework. The proposed algorithm deploys a fast load-balanced partitioning scheme along with an efficient replication-set selection algorithm, to provide fast main-memory computations of the exact AkNN results in a batch-oriented manner. Here the pruning efficiency of the Spitfire algorithm plays a pivotal role in reducing communication and response time up to an order of magnitude, compared to other state-of-the-art distributed AkNN algorithms executed in distributed main-memory.

**Keywords**: distributed system, memory distribution, spitfire algorithm, AKNN algorithm.

## 1. INTRODUCTION

In the age of smart urban and mobile environments, the mobile crowd generates and consumes massive amounts of heterogeneous data. Such streaming data may offer a wide spectrum of enhanced science and services, ranging from mobile gaming and entertainment, social networking, to emergency and crisis management services. One useful query for the mentioned services is the All kNN (AkNN) query:finding the k nearest neighbors for all moving objects.
Formally, the kNN of an object o from some dataset O,denoted as kNN(o,O), are the k objects that have the most similar attributes to o. Specifically, given objects oa6=ob6=oc, ∀ ob ∈ kNN(oa,O) and ∀ oc ∈ O−kNN(oa,O) it always holds that dist(oa, ob)≤dist(oa, oc)1. An All kNN (AkNN) query generates a kNN graph. It computes the kNN(o,O) result for every o ∈ O and has a quadratic worst-case bound. An AkNN query can alternatively be viewed as a kNN Self-Join: Given a dataset O and an integer k, the kNN Self-Join of O combines each object oa ∈ O with its k nearest neighbors from O, i.e., O⊲⊳kNNO = {(oa, ob)|oa, ob ∈ O and ob ∈ kNN(oa,O)}.
Only recently researchers have proposed algorithms for optimizing AkNN queries in such infrastructures.
The state-of-the-art AkNN algorithm consists of three phases, namely partitioning the geographic area into sub-areas, computing the kNN candidates for each sub-area that need to be replicated among servers in order to guarantee correctness and finally, computing locally the global AkNN for the objects within each sub-area taking the candidates into consideration. The given algorithm has been designed with an offline (i.e., analytic-oriented) AkNN processing scenario in mind, as opposed to an online (i.e,operational-oriented) AkNN processing scenario is an aim for in this work. The performance of can be greatly improved, by introducing an optimized partitioning and replication strategy. These improvements, theoretically and experimentally shown to be superior, are critical in dramatically reducing the AkNN query processing cost yielding results within in a few seconds, as opposed to minutes, for million-scale object scenarios.

## 2. LITERATURE SURVEY

### 2.1 EXISTING SYSTEM

It is frequently used as a classification method in machine learning or data mining. The primary application of a kNN join is k-nearest neighbour classification. Some data points are given for training, and some new unlabelled data is given for testing. The aim is to find the class label for the new points. For each unlabelled data, a kNN query on the training set will be performed to estimate its class membership. This process can be considered as a kNN join of the testing set with the training set. The kNN operation can also be used to identify similar images. To do that, description features are first extracted from images using a feature extractor technique. But the existing system lacks scalability and it is with high response time, high computation time, imbalanced loading of data.

### 2.2 PROPOSED SYSTEM

In this proposed algorithm spitfire, a distributed algorithm that solves the AkNN problem in a fast batch mode, offering both scalability and efficiency. It encapsulates a number of innovative internal components, such as:
(i) A novel linear-time partitioning process module that achieves sufficient load-balancing independent of data skewness,
(ii) A new replication process module that exploits geometric properties towards minimizing the candidates to be exchanged between servers, and
(iii) Optimizations added to the local AkNN computation are done.

## 3. DESIGN AND IMPLEMENTATION

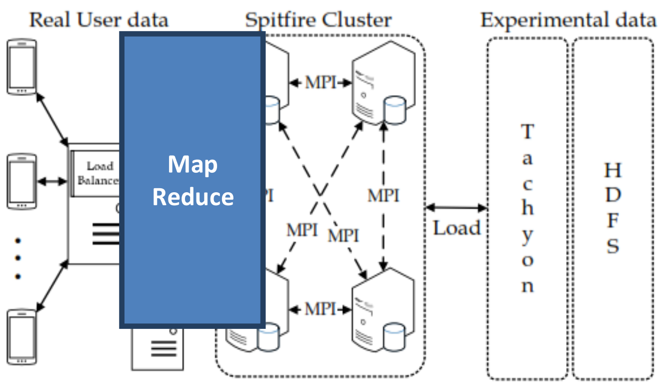

**Fig 1: Architecture**

The proposed algorithm involves the following modules

### 3.1 MODULES

#### 3.1.1 Partitioning the geographical area

- Partition the geographical area into **m** disjoint subareas with approx. equal number of objects (balancing) using hash-based equi-depth histograms
- each partition $O_i$ is assigned to a server $s_i$

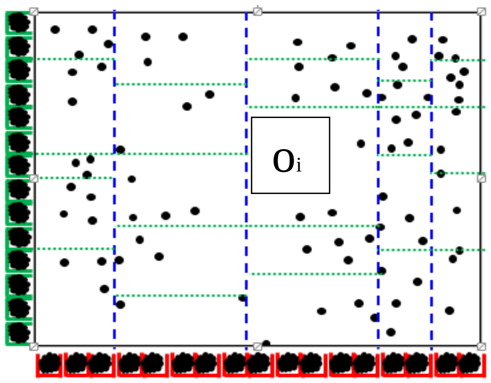

**Fig 3. Partitioning**

Steps:
1. Hash into equi-**width** buckets on x-axis
2. Group x-buckets into √**m** approx. equi-depth groups
3. For each x-group do step 1 and 2 on y-axis

#### 3.1.2 Spitfire aknn Algorithm (SAKNN)

- Given set of objects and test point p, here A is the existing object belonging to particular class.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

▶ Find distance d from p to k nearest neighbors by Euclidean distance
$\sqrt{(Px-Ax)^2+(Py-Ay)^2}$
▶ For more accuracy, use w weighted knn formula
W(A)=(max distance-distance(A))/(max distance-min distance)
▶ Assign weight value 1 to max distance
▶ Assign weight value 0 to min distance
▶ Add weight values of objects belonging to same cluster
▶ Add the test point p to the cluster having maximum weight value.

### 3.1.3 Replication

Replication is maintaining the multiple copies of the same object. As there will be geographical problems or electrical failures may lose the content where replication helps us to recover the dataset back.

- server $s_i$ computes minimum external cand. kNN set $\mathbf{EC_{ij}}$ from $O_i$ and based on border segments **B** for each neighboring server $s_j$
- transmit each $\mathbf{EC_{ij}}$ from $s_i$ to $s_j$ achieving replication factor

$$f = \frac{\sum_{ij} EC_{ij}}{n}$$

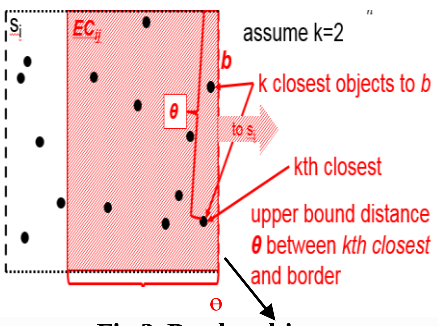

**Fig 3. Border objects**

Steps:

– Divide border into border $\sqrt{\mathbf{n}}$ segments b to minimize candidates
– Compute candidates for each b:
- Construct a Floyd Min Heap H from $O_i$ based on mindist to b -- O(n/m)
- Determine the kth closest object to b by popping k times from H
- Compute maxdist between kth closest and b called θ
- Scan $O_i$ for objects that are closer than θ to border b

– Generate $EC_{ij}$ from union of candidates for each neighboring $s_j$

### 3.1.4 Refinement

Refinement is the step helps to find the k nearest neighbour among the heap of the objects. It helps to extract k objects from the heap of objects.

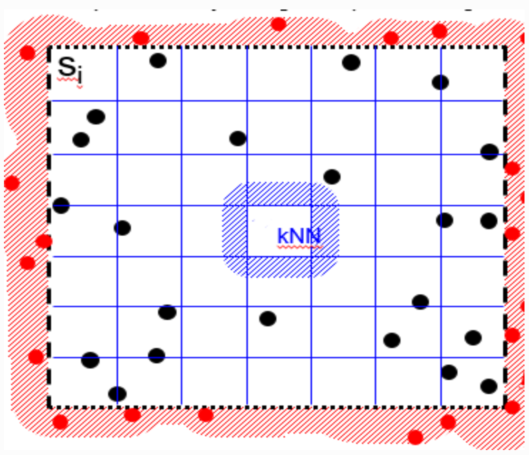

**Fig 4. Partitioning for Refinement**

DOI10.17148/IJARCCE.2017.65133

**Steps**
1. Partition using a grid
2. Find candidates per cell
3. Refinement per cell

## CONCLUSION

In this paper, a scalable and high performance distributed algorithm that solves the AkNN problem using a spitfire algorithm. This paper offers several advantages over the state-of-the-art algorithms in terms of efficient partitioning, replication and refinement. Theoretical analysis and experimental evaluation show that Spitfire outperforms existing algorithms reported in recent literature, achieving scalability both on the number of users and on the number of k nearest neighbors.

## REFERENCES

[1] F.N. Afrati, A.D. Sarma, S. Salihoglu and J.D. Ullman. "Upper and lower bounds on the cost of a map-reduce computation". In Proceedings of the 39th international conference on Very Large Data Bases (PVLDB'13), VLDB Endowment, 277–288, 2013.
[2] T. Brinkhoff. "A framework for generating network-based moving objects". Geoinformatica, Vol. 6, 153–180, 2002.
[3] P.B. Callahan. "Optimal parallel all-nearest-neighbors using the wellseparated pair decomposition". In Proceedings of the 34th IEEE Annual Foundations of Computer Science (SFCS'93), 332–340, 1993.
[4] G. Chatzimilioudis, D. Zeinalipour-Yazti, W.-C. Lee and M.D. Dika- iakos. "Continuous all k-nearest neighbor querying in smartphone networks". In Proceedings of the 13th IEEE International Conference on Mobile Data Management (MDM'12), 79–88, 2012.
[5] Y. Chen and J.M. Patel. "Efficient evaluation of all-nearest-neighbor queries". In Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE'07), 1056–1065, 2007.