Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 05, May-2019

kNN-DP:Dealing with Dataskewness in kNN-Joins Utilizing MapReduce

Archana Mohan
M.Tech Computer Science and Engineering
Believers Church Caarmel Engineering College,Perunad
Pathanamthitta, India

***Abstract*— In this examination, I found that the information skewness issue forces unfavorable effects on MapReduce-based parallel kNN-join activities running bunches. I propose an information parceling approach - called kNN-DP - to reduce load awkwardness caused by information skewness. The general objective of kNN-DP is to similarly isolate information objects into an extensive number of segments, which are handled by mappers and reducers in parallel. At the core of kNN-DP is an information parceling module, which progressively and wisely segments information to streamline kNN-join execution by stifling information skewness on Hadoop bunches. Information dividing choices to a great extent relies upon information properties (e.g., circulations), the examination of which is exceedingly costly for a gigantic measure of information. To accelerate the information property examination, I fuse an inspecting procedure to profile the information appropriation of a little example dataset speaking to huge datasets. In the wake of structure an information dividing cost model for parallel kNN-goes along with and I infer the time-multifaceted nature upper and lower limits of parallel kNN-join calculations. The cost model offers us a direction to efficiently research kNN-DP's execution. kNN-DP gets worldwide closest neighbors utilizing nearby closest neighbors. The exploratory outcomes demonstrate that kNN-DP essentially improves the execution of LSH and z-esteem while offering high extensibility and adaptability on Hadoop groups. A deduplication conspire was presented in this paper which will lessen the calculation cost.**

Keyword:- Dataskewness.deduplication,MapReduce

I. INTRODUCTION

The k-closest neighbor join (i.e., kNN-join) is a crude activity broadly received by a wide scope of datamining applications like k-implies grouping and anomaly discovery. Consolidating the kNN question and the join task, kNN-join turns into an over the top expensive task. To moderate the high overhead of kNN-join, a bunch of earlier investigations have advanced a progression of parallel kNN-join techniques utilizing MapReduce. I find that a typical impediment of the current MapReduce-based kNN-join arrangements lies in information skew issues, which lead to imbalanced outstanding task at hand among MapReduce errands. In this examination, I proposed an all-encompassing way to deal with handling the information skew issue by parceling information among hubs of a Hadoop bunch. I demonstrate that taking care of information skew can considerably improve the execution MapReduce-based kNN-join tasks. In specific, I consistently incorporate our information dividing conspire with the area touchy hashing-based (i.e., LSH [1]) furthermore, space-filling-bends based (i.e., z-esteem [2]) kNN-join calculations. I fundamentally spurred by the accompanying three perceptions to address the information skew issue in MapReducebased kNN-joins.

Perception1. A kNN-join activity consolidates each object of one dataset with its kNNs in another dataset, giving more significant question results than range joins (a.k.a., run likeness joins). kNN-joins are costly tasks, since both the closest neighbor look and the join tasks are tedious. The high overhead of kNN-join turns out to be increasingly articulated with regards to expansive datasets with multi-measurements. In the previous decade,streamlining calculations were proposed to improve kNNjoin execution [3] [4]. Prevalent advancement thoughts that help in lessening I/O and CPU costs incorporate join planning, information arranging, just as separating and decrease [5]. These basic but then effective methods improve proficiency of preparing high-dimensional information.

Perception2. An expanding number of parallel kNNjoin calculations are created to manage the quickly developing input datasets with multi-measurements (see, for instance, [6]). A larger part of customary parallel kNN-join calculations comprise of three stages, in particular, task creation, task, also, parallel undertaking execution [7]. Saving information area is an effective method for diminishing both CPU and I/O cost. Aside from traditional parallel kNN-joins, MapReduce- based kNN-joins catch much consideration in the previous few a long time [2] [8] [9] [10]. MapReduce [11] is a straightforward yet proficient parallel processing system offering high adaptability and adaptation to internal failure. Earlier examinations affirmed that MapReduce is a significant structure of preparing a huge measure of information with multi-measurements. I spurred to address execution issues in MapReduce-based kNN-join.

Perception3. I expand a charming dataskewness issue in MapReduce-based kNN-joins. I watch that current kNN-join calculations utilizing MapReduce are touchy to information attributes and information skewness. Skewed information definitely hinder parallel kNN-joins, in light of the fact that information skewness prompts imbalanced remaining burden making a few hubs an act bottleneck in MapReduce groups.The information skewness issue inspires us to propose an information dividing plan to accomplish adjusted burden in groups running MapReduce-based kNN-joins.

A. *Contributions*

The more than three recognitions move me to structure a comprehensive arrangement called kNN-DP to portion data among a Hadoop pack's datanodes to address the data skew issue. Data skewness unavoidably prompts imbalanced weight over Hadoop gatherings, thusly inside and out limiting kNNjoin execution (see also recognition 3). The data skewness ends up being continuously explained in kNN-joins when (1) input datasets R and S seek after out and out various spreads and (2) set R is liberally greater than set S. Such data skewness is facilitated by kNN-DP's data allotting framework, which applies the dynamic package limit modifying method to admirably make all of the centers likewise process kNN-participates in parallel.

The pile modifying thought of kNN-DP is through and through unique in relation to standard weight altering plans that split assignments into tinier ones and reassign a segment of the endeavors to sit processors. Rather than doling out assignments, kNN-DP hopes to modifying load through data course of action decisions making each center point similarly handle kNN-join undertakings. I lead tests to demonstrate that by managing data skewness, kNN-DP improves the execution of MapReduce-based kNN-join exercises. Data isolating decisions, as it were, depend upon data properties (e.g., courses). It is exceedingly exorbitant to get data properties of a tremendous proportion of data. To get data properties in a brief time allotment period, I propose an inspecting framework to profile the data scattering of a model dataset, which is an unobtrusive piece of a noteworthy dataset.

Like prior parallel kNN-joins, my kNN-DP gets worldwide nearest neighbors using close-by nearest neighbors. Such a conjecture course of action may be unfit to discover all the worldwide nearest neighbors, cutting down kNN-join accuracy. In solicitation to improve the kNN-join exactness, I grow the close-by data of each center marginally of abundance data, which is fastened to the head and tail of each datum square. I moreover quantitatively evaluate the impact of the kNN-DP's overabundance data strategy on the kNN-join precision.

One striking component of my data isolating arrangement is that it is symmetrical to a wide extent of MapReduce based kNN-join estimations. This segment empowers me to speedily what's more, reliably consolidate my kNN-DP with existing parallel kNN-join estimations, for instance, LSH-based [1] and z valuebased [2] kNN-joins.

Here, I first present kNN-join method, sought after by an introduction of the MapReduce framework. A deduplication plot was likewise acquainted which will help with expel the copy duplicates. Deduplication conspire is equipped for lessen the capacity tasks on account of bigger frameworks and improves the capacity use.

II. FUNDAMENTALS

A. *kNN Join*

Give us a chance to consider two datasets R and S in space Rd, where each item (e.g., r 2 R and s 2 S) is spoken to as a d-dimensional item. We measure the likeness separate between items r 2 R and s 2 S utilizing their euclidean separation d(r; s). It would be ideal if you note that the different methods for measuring likeness separation can be found in Section 4.5. Task knn(r; S) restores a lot of k closest neighbors or kNN of point r from set S.
Given article r 2 R, the kNN-join activity knnJ(R; S) of datasets R and S restores a blend set of each item r's kNN set. Therefore, we express knnJ(R; S) utilizing kNN-join task knn(r; S) as pursues. knnJ(R; S) = f(r; knn(r; S))j for all r ∈ R}: (1)

B. *MapReduce Framework*

MapReduce is a parallel programming model proposed by Google [11]. The objective of MapReduce is to disentangle the handling of extensive datasets on cheap bunch PCs. A MapReduce program commonly comprises of a couple of user defined map and decrease capacities. Hadoop is an open source programming actualizing the MapReduce processing system. Information in Hadoop are put away in a Hadoop circulated record framework, which comprises of numerous information hubs and an ace hub called namenode.

The Hadoop runtime framework builds up two procedures - JobTracker and TaskTracker. The JobTracker parts a submitted work into guide and decrease assignments, which are planned also, doled out to all accessible TaskTrackers. The TaskTrackers acknowledge and processes the relegated guide/diminish undertakings. After finishing all mappers in a Hadoop program, the Hadoop runtime framework bunches every single middle of the road result and dispatches reducers to creating last outcomes.

C. *Deduplication*

Proficient and versatile deduplication methods are required to serve the need of evacuating copied information in enormous information handling stages, for example, Hadoop. In this paper, a coordinated deduplication approach is proposed by taking the highlights of Hadoop into account and utilizing parallelism dependent on MapReduce and HBase in order to accelerate the deduplication strategy.

III. DATA PARTITIONING IN PARALLEL KNN JOINS

A. *Overview*

In this segment, I present the advancement of kNN-DP, the information parceling plan that advance kNN-joins running in the MapReduce programming structure. In the wake of advertising kNN-DP's review in the next section, then I talk about the information testing systems actualized in the information preprocessing technique. Next, I depict kNN-DP's first MapReduce work, which partitions information tests into n allotments pursued by modifying information in unequal allotments. At last, I give an depiction on the second MapReduce work that segments information in a manner to adjust load among reducers.

Profiling Sample Data. The pre-preparing system ponders tests from info datasets R and S. The profiling data on the examples catches information dissemination properties of the info datasets.The inspecting procedure executed in the preprocessing methodology improves the exactness of information apportioning requiring little to no effort.

Obtaining Data-Partition Boundaries. The first MapReduce work decides limits of information allotments. This activity plans to guarantee that each parcel's handling time intricacy is around equivalent to the best-case time unpredictability, which suggests that all the allotments are very much adjusted. The calculation of the first MapReduce work is planned utilizing the dynamic segment limit changing strategy.

Partitioning Data and Computing kNN-joins. The second MapReduce work accomplishes two objectives. To begin with, mappers in this activity parcel input information crosswise over datanodes as indicated by the limits controlled by the first MapReduce work. Second, reducers are situated to seek k closest neighbors in dataset S for r ∈ R.

a) Profiling Sample Data

A perfect information apportioning technique to improve kNN-joins should bunch objects dependent on their comparability, expecting to make various segments with equivalent burden. Whenever equalsized allotments are circulated to numerous hubs, each of which handle one parcel, the preparing time of hubs are near one another. At the end of the day, making various allotments share with comparable handling time unpredictability is a productive method for enhancing parallel kNN-join calculations. To similarly segment an expansive info information, one needs to contemplate the information's conveyance property. The overhead of profiling information property is high, since it requires arranging and examining the gigantic measure of information. To diminish the costly profiling process, I depend on a little example dataset to take after the expansive information's property. I actualize a pre-process methodology completed in the ace hub of a Hadoop group to acquire information circulation properties of enormous information from little examples. Despite the fact that there exist different testing strategies, none of these inspecting plans can be broadly connected to treat all information types. In this way, it is ostensibly evident that a commonsense route is to utilize a proper inspecting technique as per information attributes. So as to acquire information parcel limits at low processing cost, I plan the accompanying three information examining plans running on Hadoop bunches.

Simple Random Sampling. We produce a little test set from a vast info information dispersion put away on HDFS, if datasets R and S pursues an equivalent dissemination.

Heterogeneous Random Sampling. Heterogeneous information conveyance alludes to situations where information appropriations of datasets R and S are extraordinary. I proposed heterogeneous arbitrary examining to independently perform examining on R and S. In this manner, R and S have two distinctive little example sets utilizing the above straightforward irregular testing strategy.

Interval Sampling. At the point when the dispersion of datasets R and S is obscure from the earlier, I apply interim testing to profile input information. Test datasets R0 furthermore, S0 are extricated by isolating a similar number of items (i.e., "2 N) from R and S, separately. Here N is the items number of dataset R or S; " is in the range somewhere in the range of 0 and 1 (i.e., " 2 (0; 1)).

b) Obtaining Data-Partition Boundaries

The first MapReduce work in kNN-DP endeavors to separate test dataset to n equivalent gatherings by progressively modifying segment limits in unbalance gatherings. The pseudocode of this MapReduce work is point by point in Algorithm 1, which performs information parceling combined with changes utilizing MapReduce. Calculation 1 consolidates the dynamic partitionboundary modifying strategy to get problematic parcel limits in Lines 11-14 (see additionally Algorithm 2).

The mapper work in the principal work for the most part extricates highlights of a dataset with various measurements; the component extraction is actualized by rapidly ascertaining a separation between two items. I express each multi-dimensional information object as a one-dimensional esteem utilizing capacity charact(o), which might be actualized in an assortment of ways. Test executions of capacity charact(o) are the region touchy hashing-based plan [1], and the spacefilling- bends based plan [2] These two plans are regularly received in parallel kNN-join computing. I incorporate kNN-DP with LSH and z-esteem; we allude to the two kNN-DPenabled arrangements as LSH+ and z-value+. LSH+: Integrating kNN-DP and LSH-based kNNjoins.

In the LSH conspire, each item in test datasets R0 and S0 is spoken to as a hash code (i.e., one-dimensional hash esteem) by the hash work. At that point, a few items with comparable hash codes are put into a similar can, which speaks to an information parcel. Each can contains objects whose hash codes that are in a given range balanced by our kNNDP to adjust processing load among cans. z-value+: Integrating kNN-DP and z-esteem based kNN-joins. z-bend is one of the space filling bends, which maps an item in test dataset R0 or S0 to one-dimensional z esteem. The z esteems are isolated into n segments utilizing the Balance R conspire. kNNDP is connected to adjust the n parcels.

The principle objective of the Reduce work is to modify the segment limits begun by the above guide work. The parcel modification tries to adjust handling time multifaceted nature of each parcel, guaranteeing that allotments are around equivalent in size. The Reduce work makes the three strides underneath to achieve segment modifications. In the first place, test dataset R0 is arranged in a non-diminishing request of one-dimensional qualities, which are dictated by the previously mentioned guide work. The underlying parcel limits are gotten by the Balance R technique (see Lines 16-17). Second, the preparing time of each segment is evaluated utilizing time multifaceted nature examination (see Lines 19). At long last, given an information skewness-degree limit (see Section3.2), we should look at the evaluated handling time of the considerable number of segments against the best-case parcel (i.e., perfect case). Such correlations are executed in the Reduce work by contrasting the information skewness-degree and its edge. Correlation results oversee dynamic changes of information questions in a lopsidedness parcel utilizing the dynamic segment limit modifying plan (see Lines 20-24), which is portrayed in Algorithm 2.

Algorithm 1 Computing Data-Partition Boundaries

```
1: input: R0; S0;/* Two testing datasets */
2: yield: Boundary esteems;
3: work MAP (key balance, values R0 [ S0 )
4: for all (o ∈ R0 ∪ S0) do
5: o:value← charact(o);/* process o's character esteem */
6: on the off chance that (o ∈ R0) at that point/* o is an article in set R' */
7: o:f slack = Flag R/*o:f slack shows object o in R0*/
8: else
9: o:f slack = Flag S;/*o:f slack demonstrates object o in S0*/
10: end if
11: object ← o
12: emit(object; (o:value; o:f slack));
13: end for
14: end work
15: work REDUCE(key object, values (o:value, o:f slack))
16: BoundarySet ←sort(R0);/* sort dataset R0 */
17: Boundary[n]← get(BoundarySet);/* Obtain n limits from BoundarySet utilizing Balance R plot */
18: for (i=1;i<n;i++) do/*n is the quantity of partitions*/
19: Calculate the sizes jR0i j and jS0i j of R0 and S0 in I-th run;
20: if (jO(jRijlog2jSij)−Obest/ Obest≤j  T) at that point/*see Formula 5*/
21: Boundary[i] = Optimizing Boundary(Boundary[i])
/* Make limits way to deal with the best case. (see Algorithm2) */
22: else
23: emit(i;Boundary[i]_S);
24: end if
25: end for
26: end work
```

c) Dynamically Adjusting Partition Boundaries

Review I get beginning estimations of parcel limits from Calculation 1's Line 17, which has not yet comprehended the dataskewness issue. Presently we propose a calculation to decide ideal limits, easing imbalanced burden among information allotments. The pseudocode is outlined in Calculation 2, which comprises of the accompanying three stages.

Stage 1. This progression includes the quantity of items in test sets R0 and S0 in the ith segment (see Line 5), in which the quantities of articles in R0 and S0 are communicated by jR0i j and jS0 ij, separately.

Stage 2. At the point when a segment's time unpredictability is littler than that of the best case, the segment will be extended by converging with the following segment until information skewness-degree O(jRijlog2jSij)/Obest
Obest increases than limit □T (i.e., O(jRijlog2jSij)/Obest Obest □T; see Lines 6-10). Along these lines, a little segment is reached out to an extensive one. Stage 3. In a major information segment, we embrace the parallel inquiry strategy to enhance the segment's lower limit. After Stage 3 is finished, we get the underlying upper limit what's more, the enhanced lower limit, which structure a streamlined new parcel. The handling time intricacy of the new parcel is exceptionally near the perfect time unpredictability (see Lines 11-14).

Algorithm 2 Optimizing Boundary ()

```
1: input: jR0j, jS0j; Initial limit esteems: limit esteems;
2: yield: Optimized limit esteem exhibit: boundary [];
3: Boundary[n] ←Boundary esteems
4: for (i=1,j=0;i+j<n;i++) do/* n is the quantity of segments.*/
5: jR0i j, jS0i j ←count (Boundary[i-1],Boundary[i]);/* Calculate sizes jR0i j and jS0i j in a range between Boundary[i-1] and Boundary[i]. */
6: while (O(jRijlog2jSij−Obest/Obest < -T) do/*The ith parcel's time multifaceted nature is little than the perfect value.(see Equation (6))*/
7: Boundary[i]=Boundary[i+j];/* Merging segments and altering limit */
8: j++;
9: jR0i j, jS0i j count (Boundary [i-1], Boundary[i]);/* Recalculating jR0i j and jS0i j: */
10: end while
11: if (O (jRijlog2jSij)−Obest /Obest > T) at that point/*see Formula (6)*/
12: Best Binary-Search (Boundary [i-1], Boundary[i]);/* Binary look through the range between jR0i j and Best to guarantee that jR0ij  log2jS0i j  T.*/
13: Boundary[i] Best;/* Obtain the ith enhanced limit. */
14: end if
15: output (Boundary[i]);
16: end for
```

d) Partitioning Data and Computing kNN-joins.

The second MapReduce work has two duties. To begin with, this activity means to parcel information as per the limits acquired in the first MapReduce work. Second, the activity executes kNN-join tasks in parallel on a Hadoop bunch.

Algorithm 3: Data Partitioning and kNN-join Computing

```
1: input: R, S, and Boundary[n];
2: yield: kNNSet;
3: work MAP (key balance, values R0 [ S0 )
4: for all (o ∈ R ∪ S) do
5: o:value← charact(o);/* Compute character estimation of o*/
6: for (i=1; i<n; i++) do/* n is the quantity of allotments. */
7: in the event that (Boundary [i - 1] ≤ o≤value Boundary[i]) at that point
8: emit(i; (o.value; o.f lag));/* o is put in the ith parcel */
9: in the event that (o.flag=Flag S) at that point
10: Array_S[i]← o:value;
11: end if
12: end if
13: end for
```

```
14: end for
15: for (i=1;i<n-1;i++) do
16: sort(Array_S[i]);
17: RedundantMin← k least incentive in Array_S[i];
18: emit(i−1; (RedundantMin:value;RedundantMin:flag)); /* Add k excess articles in set Si-1. */
19: RedundantMax k most extreme incentive in Array S[i];
20: emit(i+1; (RedundantMax:value;RedundantMax:flag)); /* Add k excess articles in set Si+1. */
21: end for
22: end work
23: work REDUCE (key parationID, values object)
24: parse Ri and Si(Si1, Si2, : :, Sim) from (parationID, object)
25: for all (o 2 Ri) do
26: for (j=1;j<m;j++) do/* m is the quantity of items in Si. */
27: Dis[m]← distance(o:value; Sij );/* ascertain remove from article o to protest Sij*/
28: end for
29: kNN(o; S) get(Dis[m]);/* Get k least esteem */
30: emit(o; kNN(o; S));
31: end for
32: end work
```

The pseudocode of the second employment is outlined in Algorithm 3, which comprises of the accompanying five stages.

Stage 1. This progression figures the component estimation of every datum object in datasets R and S. Information items' component esteems can be utilized to think about the likenesses among these items (see Line 5).

Stage 2. Every datum object is set into a particular parcel as per the segment's limits controlled by the first MapReduce work (see the yield of Algorithm 1). This step exchanges parcel identifiers alongside a rundown of items in each segment to the Reduce work (see Lines 6-13).

Stage 3. Nearby kNN-join results are approximates of worldwide kNN-joins. To improve the exactness of rough kNN-join results, we grow each parcel of dataset S by including a head fragment and a tail section. In particular, the head fragment of the ith parcel in S contains the last k objects in the I 1th parcel; the tail fragment of the ith segment in S is involved the main k questions in the I + 1th parcel. Hence, the first and last segments of dataset S have an aggregate of k excess articles; different segments contain 2k excess articles (see Lines 15-21).

IV. CONCLUSION

In this study, I developed a data partitioning approach called kNN-DP for kNN-join. kNN-DP alleviates load imbalance incurred by the data skewness problem. kNN-DP achieves the equitable data partitioning by optimizing the partition boundaries. Specifically, kNN-DP has three salient and advanced features. First, the sampling technique is utilized to quickly capture the data distribution of a big dataset through profiling a small sample set. Second, kNNDP dynamically adjusts partition boundaries by assessing time complexity of each partition in a sample dataset. Optimized partition boundaries offer smart data-partitioning guidelines. Third, to improve the accuracy of parallel kNNjoin using MapReduce, kNN-DP employs a redundant-data strategy, which augments each node's local data by a small amount of redundant data. Also a deduplication was introduced to remove the duplicate data.

REFERENCES

[1] A. Stupar, S. Michel, and R. Schenkel, "Rankreduce-processing k-nearest neighbor queries on top of mapreduce," in Proc. 8th Workshop on Large-Scale Distributed Systems for Information Retrieval, 2010, pp. 13–18.

[2] C. Zhang, F. Li, and J. Jestes, "Efficient parallel knn joins for large data in mapreduce," in Proc. ACM 15th International Conference on Extending Database Technology, 2012, pp. 38–49.

[3] C. Yu, R. Zhang, Y. Huang, and H. Xiong, "High-dimensional knn joins with incremental updates," Geoinformatica, vol. 14, no. 1, pp. 55–82, 2010.

[4] C. Yu, B. Cui, S. Wang, and J. Su, "Efficient index-based knn join processing for high-dimensional data," Information and Software Technology, vol. 49, no. 4, pp. 332–344, 2007.

[5] C. Xia, H. Lu, B. C. Ooi, and J. Hu, "Gorder: an efficient method for knn join processing," in Proc.3th International Conference on Very Large Data Bases-Volume 30, 2004, pp. 756–767.

[6] M. Batko, C. Gennaro, and P. Zezula, "A scalable nearest neighbor search in p2p systems," in Proc. InternationalWorkshop on Databases, Information Systems, and Peer-to-Peer Computing, 2004, pp. 79–92.

[7] T. Brinkhoff, H.-P. Kriegel, and B. Seeger, "Parallel processing of spatial joins using r-trees," in Proc. IEEE Twelfth International Conference on Data Engineering, 1996, pp. 258–265.

[8] M. Jang, Y.-S. Shin, and J.-W. Chang, "A grid-based k-nearest neighbor join for large scale datasets on mapreduce," in Proc. IEEE International Conference on High Performance Computing and Communications (HPCC), 2015, pp. 888–891.

[9] G. Song, J. Rochas, F. Huet, and F. Magoules, "Solutions for processing k nearest neighbor joins for massive data on mapreduce," in Proc. 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2015, pp. 279–287.

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

[11] J. Dean and S. Ghemawat, "Mapreduce: a flexible data processing tool," Communications of the ACM, vol. 53, no. 1, pp. 72–77, 2010.

[12] M. Bouguessa and S. Wang, "Mining projected clusters in highdimensional spaces," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 4, pp. 507–522, 2009. [13] J. Zhang, S. Zhang, K. H. Chang, and X. Qin, "An outlier mining algorithm based on constrained concept lattice," International Journal of Systems Science, vol. 45, no. 5, pp. 1170–1179, 2014.

[14] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What is the nearest neighbor in high dimensional spaces?" in Proc. 26th Internat. Conference on Very Large Databases, 2000, pp. 506–515. databases,"ACM Transactions on Database Systems (TODS), vol. 24,