# Irregular Partitioning Method Based k-Nearest Neighbor Query Algorithm Using MapReduce

Qingqing Zhang[1,a], Changyun Li[2,b], Pinjie He[3,c], Xu Li[4,d] and Haojie Zou[5,d]
[1,2,3,4,5]*College of Computer and Science Hunan University of Technology 88 Taishan West Road, Tianyuan, Zhuzhou, Hunan, China.*
[a]*zhangqingqing033@163.com,* [b]*lcy469@163.com,* [c]*1027566903@qq.com*
*,* [d]*1290567217@qq.com*

**Abstract.** With the dramatic increase of available data, the process of data processing should get higher and higher performance. Most researches on k-Nearest Neighbor (kNN) query algorithm are based on the regular partitioning method which is easy to cause the imbalance of load, even influence the overall performance of the kNN query algorithm. In addition, the traditional kNN query algorithm works on single process or single machine platforms, which cannot obtain high enough efficiency when dealing with big data. Aiming at these two issues, an irregular partitioning method based kNN algorithm is presented and being executed on the distributed parallel computing platform—MapReduce as of in this paper. Experiments show that the irregular partitioning method based kNN algorithm using MapReduce can obtain much higher performance and can guarantee a very efficient query when dealing with big data.

*Keywords: big data, k-Nearest Neighbor (kNN) query algorithm, irregular partitioning method, MapReduce*

## Introduction

k-Nearest Neighbor query algorithm which can be abbreviated to kNN query algorithm is a concise and popular non-parameter classification method, which was first presented by Cover and Hart in 1967 [1] and it has been the most widely used query algorithm until now. This algorithm was used in many fields like text classification, image and space classification etc. after being proposed. And many scholars, at home or from abroad, have proposed a number of methods to improve this algorithm and all these methods could be sorted into two categories, one is just based on the kNN algorithm and the other is integrating the kNN algorithm with other algorithms. In the first category, Jing et al. presented a density-based method which clustered each class of sample data into several clusters and reduced the noise sample data and then combined some higher similar sample documents in each cluster into one document for reducing training data [2], Zhao et al. proposed an essential vector based kNN algorithm which can keep the same accuracy with before while cutting out most of the samples [3], Bhattacharya et al. put forward a new local distance function—affinity based new similarity function for kNN text classification [4], Sarma et al. substituted Gaussian distribution for linear interpolation and based it

 

on the weights to nearest neighbor [5]; while, in the second category, Ishii et al. proposed a grouping method of similar words and combined it with latent semantic analysis and kNN algorithm to obtain higher accuracy in the classification [6], Zhou et al. employed the hybrid model SVM-kNN algorithm which combined the advantages of SVM and kNN algorithms to improve the prediction accuracy of SVM [7], Jiang et al. improved kNN algorithm on the basis of Bayesian-kNN and Citation-kNN [8].

All the above improved kNN algorithms are based on the regular partitioning method which partitions text according to the projected area in a two-dimensional coordinate (only take two-dimensional coordinate into account here) into several small texts of the same area, text in each grid is a small text fragment which is given processors of the same number, kNN algorithm is implemented on the basis of these text fragments. The regular partitioning method based kNN algorithm is simple and tending to realize, but it may cause the issue of computational load imbalance due to the unevenly distributed data, which will markedly influence the comprehensive performance of kNN algorithm especially when in the face of big data.

On the other hand, the traditional kNN algorithm is executed on single process or single machine platforms which cannot guarantee an efficient data processing under the circumstance of big data. However, distributed parallel computing platforms have been widely used in dealing with big data which can handle data in parallel to get better computing performance. Hence, executing kNN algorithms on distributed parallel computing platforms will be a very good way to increase the efficiency of the kNN algorithm.

As a reaction to these two issues, this paper presents an irregular partitioning method based kNN algorithm to avoid the data skew [9] problem and performs it on MapReduce, one of the most popular distributed parallel computing platforms, to improve the data processing ability, and this improved kNN algorithm implemented on the MapReduce platform gives us a very good way to process data efficiently especially when dealing with big data.

The rest of this paper is organized as follows. Section 2 introduces the related work, the k-Nearest Neighbor (kNN) query algorithm and the distributed parallel computing platform: MapReduce. Section 3 explains the irregular partitioning method based kNN algorithm in detail, and executes the improved kNN algorithm on MapReduce. Experimental results and our analysis of them are presented in Section 4 and section 5 presents conclusions.

## Related Work

**The k-Nearest Neighbor Query Algorithm.** The main idea of kNN algorithm is to predict the category of the given text X according to the known category of the *n* training texts ($T_1,T_2,\ldots,T_n$), and select out *k (k>=1)* texts which are most closest or similar to the given text X, of which coordinate values can be expressed as ($C_1,C_2,\ldots,C_n$), similarity values in this text are measured taking use of Euclidean distance. Suppose coordinate values of $T_1$ are ($W_1,W_2,\ldots,W_n$) the similarity value between X and $T_1$ can be computed by:

$$\cos(X,T)=\frac{\sum_{i=1}^{n}(C_i*W_i)}{(\sum_{i=1}^{n}C_i^2*\sum_{i=1}^{n}W_i^2)^{\frac{1}{2}}}. \quad (1)$$

The bigger the $\cos(X,T)$, the more similar the two texts are.

**MapReduce Framework.** The concise and abstract distributed parallel computing model— MapReduce [10] was first presented by Google in 2004 and then being used to simplify large-scale data processing, and it has been widely used in this field today [11,12], for example, the widely used open source implementation Hadoop [13]. MapReduce runs on the basis of Hadoop—the Hadoop Distributed File System (HDFS) [14]. The motive of MapReduce, which has two user defined functions—the Map function and the Reduce function, is to deal with files from HDFS and express them in key/value pairs, then these key/value pairs will be input to Map functions and output key/value pairs after a series of processing as intermediate results. These intermediate results from Map functions will be send to Reduce functions after the shuffle operation executed by MapReduce, final results will be output in key/value pairs after a series of operations executed by user defined Reduce functions.

This makes only take Map functions and Reduce functions into consideration by programmers when dealing with big data in parallel into reality. Benefit from the thinking way of "dividing and dealing" that divide one big file into several small files, the complexity of data processing is reduced significantly and the processing efficiency is improved substantially. So it will be a very good substitute for the single process or single machine platform of the kNN algorithm in big data time.

## The Irregular Partitioning Method Based kNN Algorithm

The main idea of the kNN algorithm based on the irregular partitioning method is: Assume the given text to be classified is *t*, the given dataset *D* is formed by data points in Euclidean space, there are $N_x$ grids from east to west and $N_y$ grids from south to north in a two-dimensional coordinate which *D* projects on, and also *M* processors in work. Divide these *M* processors into *G* groups and group *q* has $G_q$ processors [15].

$$G=[\sqrt{M\frac{N_x}{N_y}}]. \quad (2),$$

$$G_q = \begin{cases} [\frac{M}{G}]+1 & q \le Mod\left(M,G\right) \\ [\frac{M}{G}] & q > Mod\left(M,G\right) \end{cases}, \qquad q \in \left[1,G\right]. \qquad (3).$$

Express one point $p$ in D by *(p.i,p.j)*, the computing load of it by $L_{ij}$ and the total load by $L$:

$$L_{ij} = \begin{cases} 1, \sqrt{\left(i-i_0\right)^2+\left(j-j_0\right)^2} > 10 \\ 10, \sqrt{\left(i-i_0\right)^2+\left(j-j_0\right)^2} \le 10 \end{cases}, \qquad i_0=N_x/2,\ j_0=N_y/2. \qquad (4),$$

$$L = \sum_{i,j\in D} L_{ij}. \qquad (5).$$

Suppose $L_q = \sum_{n=1}^{q} \frac{L}{M} G_n$, $n \in \left[1, G-1\right]$. (6), and take $L_{min}$ as the minimum $L_{ij}$ in $D$. Divide $D$ into G regions $G_q$:

```
Begin
  q = 1, j = N_y
  while j ≥ 1
    i = 0
    while (i++) ≤ N_x
      mark(i, j) with q and calculate Σ_{i,j∈D} L_ij
      if |Σ_{i,j∈D} L_ij − L_q| = L_min
        q = q + 1
      else
        break
    j = j − 1
End
```

Fig. 1 Pseudo-code for dividing regions.

Fig.1. describes the process of dividing $D$ into $G$ areas which number is expressed by $q$ according to the load of each area. The outside looping variable is j which varies from $N_y$ to 1, while the inside variable is i which varies from 1 to $N_x$, if one point belongs to this area, then calculate out its $L = \sum_{i,j\in D} L_{ij}$ and compare it with $L_q = \sum_{n=1}^{q} \frac{L}{M} G_n$, only when the difference is equal to $L_{min}$ can $D$ be divided into $G$ areas according to these points.

## The Improved kNN Algorithm in MapReduce

The situation of big data today generates demands of higher efficiency and better performance of the popular kNN algorithm, the above irregular partitioning method partitions the whole text area into several small text areas and number them respectively, which fits very well with the operational mechanism that HDFS divide one big area into several small areas first and then pass them to the most popular parallel computing platform—MapReduce, so run the irregular partitioning method based kNN algorithm on MapReduce can significantly improve the comprehensive performance of kNN algorithm when dealing with big data.

Rewrite the getSplits() and the getRecordReader() method in the Map function according to the pseudo-codes is introduced above, the basic thought of running the irregular partitioning method based kNN algorithm on MapReduce is: users upload files to HDFS and a Map function downloads one text area each time, then the modified Map function divides it into several regions. Assume one small region in a mapper is $D_q$ which number is $q$, and the input *key/value* pair can be $(q, D_q)$ which key is $q$ and value is $D_q$; this region will be transformed into a specific format file and the similarity value $s_q$ between this text and the given text $t$ will be calculated out, using the former formula (1), after traversing the whole text. Set a collection $S$ with $t$ and $s_q$, which form is $S(t,s_q)$, then, take q as the key and $S(t,s_q)$ as the value, output the intermediate *key/value* pair $(q, S(t,s_q))$, which is the result of Map function as well as the input of Reduce function. After reading in, the Reduce function sorts $s_q$ with the same key in ascending order and select out the first $k$ splits, MapReduce shuffles all the splits from Map functions after that selection and sorts them in ascending order, then, select out the first $k$ splits among all of them. Set a collection $O$ with the $D_q$ which corresponds to the $s_q$ from the selected $k$ splits which can be express with $O(D_1, D_2,\ldots, D_k)$, and count the number of times each $D_q$ appears. The category $m$ i.e. the subscript number of $D$ is the category of the given text $t$. Take $m$ as the key and $S(t,s_m)$ as the value, the output *key/value* pair of the Reduce function is $(m, S(t,s_m))$, which is just the final result. The following figure gives the process:

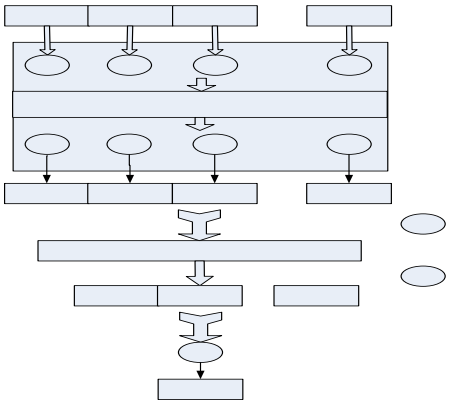

Fig. 2 Procedure of the improved kNN algorithm in MapReduce.

## Experiments and Analysis

In this section, the comparative experimental results for the traditional kNN algorithm and the improved kNN algorithm based on different platforms are shown.

**Experimental Environment.** The experimental platform is composed of a single server as the master node and two clients as slaver nodes which are both equipped with 4.00GHz CPU, 8G RAM and 50G hard disk. Set the number of mapper and reducer in each slaver node as 4 respectively. The operating system is Cent OS 6.5, the version of Hadoop is Hadoop-0.20.1.

Data sets used in this experiment are the standard data set $T_1$ and the simulated data set $T_2$. $T_1$ is the Thyroid Disease data set from UCI [16], which consists of 7200 samples, while, $T_2$ is dichotomous data generated by the random function in MATLAB:

$$R = gamrnd\left(A, B, v\right) \quad (7).$$

**Results and Analysis.** Firstly, execute the traditional kNN algorithm and the improved kNN algorithm raised in this paper on single machine and the MapReduce platform, respectively, using data set $T_1$, and results of them are shown in fig.3.The fact that time consumed by the traditional kNN algorithm executed on single machine platform which is 21 minutes is much more than the improved kNN algorithm on the same platform which is 18 minutes shows that the improved kNN algorithm can consume less time than the traditional one under the same circumstance. While, time consumed by the traditional kNN algorithm executed on MapReduce, such as it takes 20 minutes when there are 4 nodes, is much more than the improved kNN algorithm executed on MapReduce, since the improved kNN algorithm only consumed 16 minutes with the same nodes, this phenomenon points out that the improved kNN algorithm implemented on MapReduce gets higher efficiency than both the improved kNN algorithm executed on single machine platform and the traditional kNN algorithm executed on the above two platforms.

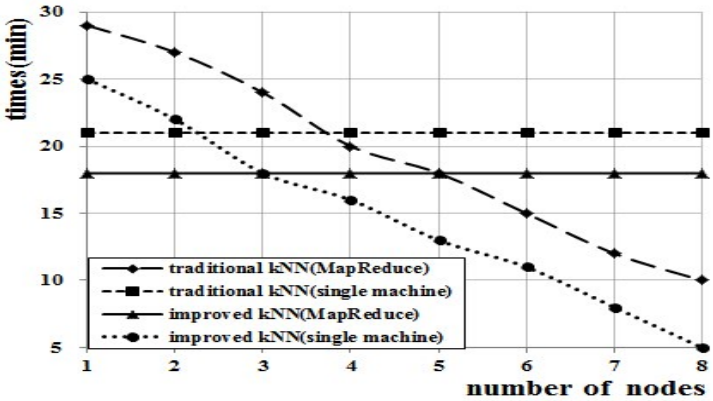


Fig. 3 Time costs of the traditional and improved kNN algorithm on MapReduce and single machine.

Secondly, Speedup, which is a very momentous measure of the parallel performance and efficiency of programs, can be expressed as:

$$Speedup = \frac{\sum_{i,j \in D} L_{ij}}{Max\left(\sum_{i,j \in D} L_{ij}\right)} \tag{8}$$

This paper executes the traditional kNN algorithm and the improved kNN algorithm under data set $T_1$and data set $T_2$ on the MapReduce platform, respectively.

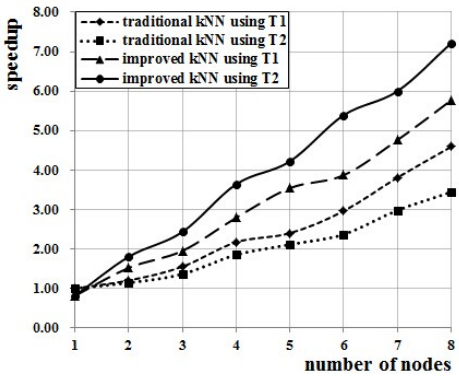


Fig. 4 Speedup of the traditional and improved kNN algorithm on data set $T_1$ and data set $T_2$.

The result, as is shown in fig.4, shows a fact that the speedup of the improved kNN algorithm using MapReduce is higher than that of the traditional kNN algorithm no matter on data set $T_1$ or on data set $T_2$, for example, speedups are

2.37, 2.96, 3.87, 5.39 when there are six nodes, the first two number representing speedup of the traditional kNN algorithm using $T_1$ and $T_2$, the latter two—speedup of the improved kNN algorithm using $T_1$ and $T_2$, which exactly demonstrates that the irregular partitioning method based kNN algorithm can perform much better no matter when using the simulated data set or using the generated data set than the traditional kNN algorithm on the MapReduce platform.

Finally, precision of this query algorithm is different when *k* is different, however, even with the same *k*, precision can still be different on different data sets. The following fig.5 shows the precision of the traditional kNN algorithm and the improved kNN algorithm on data set $T_1$ and data set $T_2$ with different k concerned this paper.

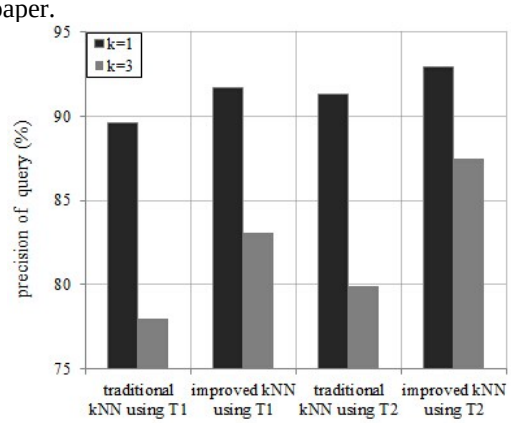


Fig. 5 The precision of results of the traditional and improved kNN algorithms.

Two different values of *k* (*k=1* and *k=3*) are taken here, it can be seen that precisions of the improved kNN algorithm using both $T_1$ and $T_2$are higher than the traditional kNN algorithm using the two same data sets, take *k=3* for example, precisions of the improved kNN algorithm and the traditional kNN algorithm using $T_1$ and $T_2$ are 83.1% 87.5% and 78% 79.9%, respectively. That is to say, precision of the improved kNN algorithm is higher than that of the traditional kNN algorithm whatever *k* takes.

Taken together, the irregular partitioning method based kNN algorithm can obtain much better comprehensive performance than the traditional kNN algorithm using MapReduce especially when facing with big data.

## Conclusion

In order to improve the efficiency of processing more and more cumulated data, this paper proposes an enhancement method for the basic and important query algorithm in spatial database field—the k-Nearest Neighbor (kNN) algorithm with substituting the irregular partitioning method for the regular partitioning method and executes it on the most popular programming model for distributed parallel computing—MapReduce. The improved kNN algorithm have gained good performance in balancing load as well as avoiding the data skew problem, nevertheless, more attention should be invited to perfect the calculation of the

computing load, at the same time, the irregular partitioning method based kNN algorithm using MapReduce should be studied in high dimensions to improve data manipulation capabilities and should be realized on larger size platforms to enhance data processing abilities in future.

**Acknowledgements**

This work was financially supported by the state ministry of science and technology (2013BAJ10B14-5) and the natural science foundation of hunan university of technology (2014hzx19).

**References**

[1] Cover T. and Hart P.: IEEE Transactions on Information Theory, Vol.13 (1967), p.21-27.
[2] Yongxia Jing, Heping Gou and Yaling Zhu: Computational and Information Sciences (ICCIS), Vol.6 (2013), p.972-975.
[3] Weidong Zhao, Shuanglin Tang and Weihui Dai: Electronics & Electrical Engineering, Vol.7 (2012), p.119-122.
[4] Gautam Bhattacharya, Koushik Ghosh and Ananda S. Chowdhury: Pattern Recognition Letters, Vol.3 (2012), p.356-363.
[5] T. Hitendra Sarma, P. Viswanath, D. Sai Koti Reddy and S. Sri Raghava: Recent Advances in Intelligent Computational Systems (RAICS), Vol.9 (2011), p.227-231.
[6] Naohiro Ishii, Takeshi Murai, Takahiro Yamada, Yongguang Bao and Susumu Suzuki: IEEE Computer Society Washington, DC, USA, Vol.10 (2011), p.393-400.
[7] Hanhai Zhou, Jinjin Wang, Jiadong Wu, Long Zhang, Peng Lei and Xiaoyun Chen: Computational Intelligence and Security (CIS), Vol.12 (2013), p.174 - 177.
[8] Liangxiao Jiang, Zhihua Cai, Dianhong Wang and Harry Zhang: International Journal of Machine Learning and Cybernetics, Vol.5 (2013), p.193-199.
[9] Lars Kolb and Erhard Rahm: Datenbank-Spektrum, Vol.13 (2013), p.23-32.
[10] Ralf Lämmel: Science of computer programming, Vol.68 (2007), p.1-30.
[11] Feng Li, Beng Chin Ooi, M. Tamer Özsu and Sai Wu: ACM Computing Surveys (CSUR), Vol.3 (2014), p.124-132.
[12] Anish Das Sarma, Foto N. Afrati, Semih Salihoglu and Jeffrey D. Ullman: Proceedings of the VLDB Endowment, Vol.4 (2013), p.277-288.
[13] Hadoop, http://hadoop.apache.org/.
[14] Tom White, in: Hadoop: The Definitive Guide, edited by Tom White, Publications/O'Reilly Media Inc, USA (2010), in press.
[15] Zhiyan Jin and Dingxing Wang: Acta Meteorologica Sinica, Vol.60 (2002), p.150-156, in Chinese.
[16] UCI Machine Learning Repository; http://archive.ics.uci.edu/ml/datasets.html.