Chapter 11
Voronoi-Based kNN Queries Using K-Means Clustering in MapReduce

Wei Yan
Liaoning University, China

ABSTRACT

The kNN queries are special type of queries for massive spatial big data. The k-nearest neighbor queries (kNN queries), designed to find k nearest neighbors from a dataset S for every point in another dataset R, are useful tools widely adopted by many applications including knowledge discovery, data mining, and spatial databases. In cloud computing environments, MapReduce programming model is a well-accepted framework for data-intensive application over clusters of computers. This chapter proposes a method of kNN queries based on Voronoi diagram-based partitioning using k-means clusters in MapReduce programming model. Firstly, this chapter proposes a Voronoi diagram-based partitioning approach for massive spatial big data. Then, this chapter presents a k-means clustering approach for the object points based on Voronoi diagram. Furthermore, this chapter proposes a parallel algorithm for processing massive spatial big data using kNN queries based on k-means clusters in MapReduce programming model. Finally, extensive experiments demonstrate the efficiency of the proposed approach.

INTRODUCTION

The k nearest neighbor query (kNN query) is a classical problem that has been extensively studied, due to its many important applications, such as knowledge discovery, data mining, and spatial databases. The k nearest neighbor query (kNN) is a special type of query that is k nearest neighbors from points in S for each query point r in dataset R. The kNN query typically serves as a primitive operation and is widely used in spatial databases. The basic idea of implementing kNN queries is to perform a pairwise computation of distance between each object point in S and each object point in R. The computational complexity of such pairwise calculation is $O(|R| \times |S|)$. Then, finding the k nearest neighbors in S for every r in R corresponds to sorting the computational distances, and easily leads to a complexity of $|S|$

DOI: 10.4018/978-1-5225-8446-9.ch011

$\times$ log|S|. Therefore, a lot of research works have been dedicated to improve the efficiency of the *k*NN queries and reduce the computational complexity of the *k*NN queries (Jagadish et al. 2005) (Bohm et al. 2004) (Ciaccia et al. 1997) (Yu et al. 2010). With the rapid growth of spatial data, the parallel *k*NN query has become a challenging task.

MapReduce is a parallel processing framework that uses parallel and distributed patterns to process massive data sets. The MapReduce programming model provides good scalability, flexibility and fault tolerance. MapReduce was first introduced by Google (Dean et al. 2008) and ran on the Hadoop clusters, which is an open source framework. MapReduce provides a programming model to process the large scale data sets, which can be distributed easily. The MapReduce programming framework can install on computational clusters and automatically distribute a work job on clusters of machines. Therefore, MapReduce programming model becomes an ideal framework of processing *k*NN queries over massive spatial datasets. This chapter proposes a method of parallel *k*NN queries based on k-means clusters using MapReduce programming model.

Now, lots of researches (Yao et al. 2010) have been devoted to improve the performance of *k*NN query algorithms. However, all these approaches are performed on a single, centralized server. In single machine, the computational capability and storage are limited, and its efficiency is low. How to perform the *k*NN query on parallel machines is an important issue in cloud computing environments. ALL the existing work has concentrated on the spatial databases based on the centralized paradigm. Xia et al. (2004) proposed a novel *k*NN-join algorithm, called the Gorder *k*NN join method. Gorder is a block nested loop join method that exploits sorting, join scheduling and distance computation filtering and reduction to reduce both I/O and CPU costs. It is simple and yet efficient, and handles high-dimensional data efficiently. However, the system of centralized server will eventually suffer from performance deterioration as the size of the dataset increases. A solution is to consider the parallel query processing in distributed cloud computing environment. However, an efficient parallel *k*NN queries in MapReduce programming framework are challenging tasks. Firstly, the classical query processing algorithms need to be redesigned in MapReduce programming framework. Second, the strategies of data partitioning and distribution need to be designed also in parallel programming model. For this purpose, we improve previous implementations of kNN queries in MapReduce programming framework, focusing on the different steps involved in Map and Reduce phases.

Parallel spatial query processing has been studied in parallel database, cluster systems as well as cloud computing platform. In cloud computing environments, a large part of data-processing using MapReduce (Dean et al. 2004) programming model runs extensively on Hadoop. The MapReduce programming model provides a powerful parallel and distributed computing paradigm. For such data intensive applications, the MapReduce programming framework has applied as a platform for big data processing. Cui et al. (2014) addressed the problems of processing large-scale data using *k*-means clustering algorithm and proposed a novel processing model in MapReduce to eliminate the iteration dependence and obtain high performance. A data structure that is extremely efficient in exploring a local neighborhood in a geometric space is Voronoi diagram (Okabe et al. 2000). Given a set of points, a general Voronoi diagram uniquely partitions the space into disjoint regions. The region corresponding to a point *p* covers the points in space that are closer to *p* than to any other point. Recently, the parallel *k*NN queries in MapReduce programming framework, such as H-zkNNJ (Zhang et al. 2012), H-BRJ (Zhang et al. 2012) and PGBJ (Lu et al. 2012), were proposed. The method PGBJ is shown to outperform H-zkNNJ and H-BRJ since PGBJ performs the early pruning of object points for non-*k*NN queries. However, the pruning power of method PGBJ is not effective when the size of datasets becomes very large.