Journal of Network and Computer Applications 44 (2014) 172–182

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Scalable nearest neighbor query processing based on Inverted Grid Index

Changqing Ji [a,b], Zhiyang Li [a], Wenyu Qu [a,*], Yujie Xu [a], Yuanyuan Li [a,c]

[a] School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China
[b] School of Physical Science and Technology, Dalian University, Dalian 116622, China
[c] School of Software, Dalian Jiaotong University, Dalian 116028, China

ARTICLE INFO

Article history:
Received 9 January 2014
Received in revised form
11 May 2014
Accepted 22 May 2014
Available online 10 June 2014

Keywords:
Spatial database
NN query
Inverted Grid Index
Big data

ABSTRACT

With the increasing availability of Location-Based Services (LBS) and mobile internet, the amount of spatial data is growing larger. It poses new requirements and challenges for distributed index and query processing on large scale spatial data. A scalable and distributed spatial data index is important for the effective Nearest Neighbor (NN) query. There are several approaches that implement distributed indices and NN query processing with MapReduce, such as R-tree and Voronoi-based index. However, R-tree is unsuitable for parallelization and Voronoi requires extra computation for localization or local index reconstruction. In this paper, we investigate how to perform NN queries in a distributed environment. Firstly, we present distributed approaches that construct a novel distributed spatial data index: Inverted Grid Index, which is a combination of inverted index and grid partition. Secondly, we illustrate the implementations of two typical applications: distributed *k* Nearest Neighbor (kNN) and Reverse Nearest Neighbor (RNN) queries which are based on our index structure under cloud computing environment. Finally, we evaluate the effectiveness of our algorithms with extensive experiments using both real and synthetic data sets. Our experiments demonstrate that the time of constructing index structure decreases almost linearly as the number of cluster nodes increases. The results also demonstrate the efficiency and scalability of our NN query algorithms based on Inverted Grid Index.

1. Introduction

Large-scale data analysis is the core of main enterprises and scientific research. In recent years, with the development of LBS and mobile internet, the amount of geospatial data is growing rapidly. Due to increasing volume of spatial data, there have many problems for traditional indexing mechanisms which usually assume an in-memory index or optimize for local disk access. Processing spatial analysis and query lead to frequent disk access using traditional methods. Thus it becomes essential to provide efficient index and query methods for spatiotemporal databases.

Nearest Neighbors (NN) algorithms draw a lot of attention in recent years. Typical spatial NN queries algorithms include *k* Nearest Neighbors (kNN) and Reverse Nearest Neighbors (RNN), and so on. kNN query retrieves *k* objects that lie closest to the query point *q* among a given set of objects *P*. RNN query is to find all the objects whose nearest neighbor is the given query point. Both kNN and RNN queries are popular in intelligent navigation, traffic control, profile based marketing and other areas. An example of large-scale kNN query is to find some closest restaurants at the New York airport initiated by a GPS device. It is a RNN problem if the restaurant has a marketing application in which the issue is to determine the business impact of restaurants to each other.

Although NN has been studied extensively, almost all the existing works are based on the centralized paradigm that NN is performed on a single centralized server, such as Korn and Muthukrishnan (2000), Mokbel et al. (2004) and Tao et al. (2004). Early research on parallel kNN algorithms in a distributed shared-nothing multiprocessor environment such as SINA (Mokbel et al., 2004), introduces the scalable incremental hash-based algorithm to evaluate a set of concurrent continuous spatial-temporal queries. TPL algorithms (Tao et al., 2004) use half-space pruning for exact RNN processing. These methods join in-memory index with in-disk algorithm to achieve scalability and efficiency. Because of the limited computational capability and storage resources of a single machine, these algorithms will eventually suffer from performance deterioration as the dataset grows larger, especially for the high-dimensional datasets. So, most existing methods of NN query cannot deal with the large-scale dataset.

The parallel and distributed processing seems a good solution to these problems. MapReduce (Dean and Ghemawat, 2008) is a

* Corresponding author.
E-mail address: wenyu@dlmu.edu.cn (W. Qu).

http://dx.doi.org/10.1016/j.jnca.2014.05.010
1084-8045/

widely used parallel programming model and computation platform on cloud computing environment. It is very easy to develop scalable distributed programs to process data-intensive applications on clusters of commodity machines. MapReduce has emerged as one of the most widely used parallel computing platforms for processing data on terabyte and petabyte scales. Soon after its birth, MapReduce gains a lot of popularity for its simplicity, flexibility, fault tolerance and scalability. It becomes an ideal framework for processing big data operations. We will show later that existing methods of NN query cannot be applied or extended into MapReduce easily. This mainly lies in two aspects: lacking of both appropriate distributed index and effective large-scale query processing methods.

There are several existing approaches that construct spatial indices distributed, such as parallel building R-tree and Voronoi-based indices in MapReduce paradigm. Ariel Cary et al. (Papadopoulos and Manolopoulos, 1997) proposed the bulk construction of R-trees and aerial image quality computation with MapReduce model. However, tree-based indices (Cary et al., 2009) do not scale due to the traditional top-down search that overloads the nodes near the tree root, and fails to provide full decentralization. Complex structure makes it difficult to be paralleled and assemble boundary. Hence, there is considerable overhead of solving node boundary problems. Moreover, the authors only presented the distributed method of constructing index, but did not explain how to process the queries in their paper. Akdogan et al. (Akdogan et al., 2010) proposed the approach of creating a spatial index based on Voronoi diagram for given data objects only in 2D space and enabled efficient processing of a wide range of geospatial queries. But it needs to build index and execute query for the enquiry node while processing NN queries. As a result, when dealing with random queries, there would be extra computation for localization or local index reconstruction.

Previous work on designing NN query algorithms with MapReduce is generally based on the following approaches. The input dataset is partitioned into groups and each group is assigned to a unique machine that performs some computation. Then, they collect the results of the previous step from the different machines into a single machine, which performs some computation and returns the final solution. We can also use the similar approach for the NN query problems. However, these methods can not improve the suitability of NN on MapReduce model. Most of the existing NN query algorithms have inherently sequential characteristics. So, there are still some difficulties to make the NN algorithms parallel using MapReuce model.

To our knowledge, our work is the first attempt in designing a distributed inverted spatial grid index in the cloud computing context and using it to process spatial kNN and RNN queries with MapReduce. Our contributions are as follows:

(1) We propose a distributed Inverted Grid Index for given data objects in multi-dimensional space, which meets the standard of spatial index: dynamic and simple. Since grid-based index is simple (Cheema et al., 2007), it can be easily updated and distributed. Instead of listing all the objects in the forward index, the Inverted Grid Index structure is developed which lists the objects per cell. The loose coupling and shared nothing architecture of Inverted Grid Index scales well.
(2) We implement the Inverted Grid Index with MapReduce and propose two scalable NN query algorithms, which are based on our index structure. The implementation designs the mapper and reducer jobs, and we do not change the existing MapReduce framework.
(3) We also conduct a lot of experiments about constructing the Inverted Grid Index and processing NN queries to verify the scalability of our proposed index structure and NN queries performance. The results show that our index constructing time is over 25 percent less than R-tree and Voronoi-based index and the scalability also outperforms the other two index structures when compute nodes more than four. Our NN query algorithms are at least three times faster than Voronoi-based query processing.

This paper extends the earlier published conference paper (Ji et al., 2012) in several substantial aspects. First, we add more details about the background, motivation and the related work. Second, detailed descriptions of scalable RNN query process have been added based on inverted grid index. Finally, we added more experiments to evaluate the performance of our proposed algorithms.

This paper is organized as follows. Section 2 presents related work. Section 3 proposes the Inverted Grid Index structure and its construction with MapReduce. Section 4 presents the NN queries processing algorithm: ParallelCircleTrip and MRRNN based on Inverted Grid Index. Section 5 describes the experimental results and performance evaluation. Finally, Section 6 concludes our work.

2. Related work

Processing large-scale spatial index and NN queries have been studied intensively. The key challenge lies in the large number of location information that must be managed by an appropriate indexing structure and efficient query processing algorithms.

Index has a critical impact on the large scale data access. There are several distributed spatial index approaches like R-tree (Cary et al., 2009) and Voronoi-based index (Akdogan et al., 2010). However, the hierarchical indices like R-tree are structurally unsuitable for MapReduce. Since this hierarchical structure is complex and serial, it does not scale well and fails to provide full decentralization. Voronoi-based index (Akdogan et al., 2010) can be parallelized, but the location of an object in Voronoi-based index takes extra time and updating of index is also more complex. Chen and Tu proposed D-Stream (Chen and Tu, 2007), a grid-based partition algorithm. It is more suitable for parallelization. Inverted Index is widely used in text retrieval (Zobel et al., 1998), which utilizes limited index entries to index unlimited data terms to allow fast full-text searches. Inspired by the Inverted Index, the bag of visual words model was proposed for image retrieval. Heng Qi et al. proposed the AM-LCF feature vector and kernel (Qi et al., 2012) to measure the similarities between sets of feature vectors.

NN queries have received intensive attention in spatial database community in the past decade (Böhm and Krebs, 2004; Henrich, 1994; Mokbel et al., 2004; Cheema et al., 2007; Papadopoulos and Manolopoulos, 1997). In an earlier researches, a snapshot kNN (Henrich, 1994) query is to find the kNN from a static dataset. Some similarities with kNN processing can be found in recent years. CircularTrip (Cheema et al., 2007) and Grid-Tree (Hasan et al., 2010) studied the problem of continuous kNN query with an in-memory grid index. Taking advantage of grid index, its data access method is simple and efficient. Furthermore, it can be distributed. Under restrictions of node memory, in-memory index is only able to deal with small-scale dataset instead of large-scale dataset. The locality-sensitive hashing (LSH) (Stupar et al., 2010) method only works for data in very high dimensions. Korn et al. firstly (Korn and Muthukrishnan, 2000) defined the RNN queries and provided a large number of applications. The cost of brute-force approach is expensive, which scans and computes the whole input dataset objects. Methods proposed by KM (Korn and Muthukrishnan, 2000) and TPL (Tao et al., 2004) are structurally unsuitable for distribution.