STSM Title: Spatio-temporal algorithms in MapReduce frameworks

Action: IC0903 – Knowledge Discovery from Moving Objects (MOVE)

Applicant: Diego Seco Naveiras
Database Lab. Computer Science Department. University of A Coruña
15071, A Coruña (Spain)

Host: Apostolos Papadopoulos
Data Engineering Lab. Department of Informatics. Aristotle University
54124 Thessaloniki, Hellas (Greece)

Scientific Report

1. Purpose of the STSM

The purpose of this STSM is defined by the following two facts. First, given that trajectory data tends to be massive and that queries are becoming increasingly complex, it is necessary to define algorithms and data structures that improve the execution efficiency both in terms of time and space. Second, the recent availability of distributed computing infrastructures (e.g., computer clusters) and platforms that can be used by means of a Web service paradigm (e.g., Amazon Elastic Compute Cloud, EC2) has changed the way that large computing tasks are performed. Furthermore, some commercial and open source MapReduce frameworks have been defined and implemented (e.g., Apache Hadoop). These frameworks define a common conceptual paradigm to approach highly distributable problems and infrastructures.

However, little work had been done to research how trajectory queries can be implemented using MapReduce frameworks. Particularly, which data structures are more suitable, which query types can benefit from these frameworks, and how algorithms must be adapted. Therefore, the purpose of this STSM was to research the state of the art, to define a detailed work plan, and to produce the first data structures and algorithms on this subject.

2. Description of the work carried out during the STSM

Before the STSM, some work had been done by researchers from the home and host institutions in order to prepare a preliminary study of the state of the art. Other researchers (Sergio Ilarri and Dragan Stojanovic) also collaborated on this task.
Based on this study of the state of the art, the first task accomplished during the STSM was the identification of interesting problems related with the purpose of the mission. These problems include the adaptation to MapReduce frameworks of the following tasks: management of trajectory data, effective sampling of trajectories, construction and bulk-loading of spatio-temporal indexes, spatio-temporal joins, etc.
To the best of our knowledge few work had been done on these problems. In [CLUSTER'09] authors present an algorithm to perform spatial joins on Map-Reduce. This paper is an adaptation to the MapReduce framework of the PPBSM algorithm by Patel and DeWitt [SIGMOD'96] for parallelizing spatial joins. We start with this algorithm as a baseline and study how it can be extended for trajectories of moving objects. As our main results are in this topic, we present them in more detail in the next section. We conclude this brief summary of the state of the art with other references relevant to our work. In [CloudDB'09] authors propose an extension of the MapReduce framework to handle trajectories. Unlike their proposal, our algorithms can be implemented in

Hadoop without any modification, which makes them suitable for practical applications. In [SSDBM'09], authors study the bulk-loading of R-trees and aerial image quality computation in MapReduce. Finally, although not directly related with spatio-temporal data, in [EDBT'10] and [SIGMOD'11] some interesting ideas about how to perform and optimize joins in a MapReduce environment are presented.
The main problem studied in this STSM is the spatial-join in MapReduce of a set of trajectories and a set of spatial objects. This problem is the natural extension of the work in [CLUSTER'09] and it has many applications in real life. For example, to complement trajectories with semantic content or to detect trajectories violating restricted areas. Our main results in this topic are described in the next section.
Finally, in this STSM we defined a plan to continue with this work and for future collaborations. This plan is described in Section 4.

3. Description of the main results obtained

Let us first formalize the problem of our study. The input is defined by a collection of spatial regions R and a collection of trajectories T. Given these two collections our goal is to compute the spatial-join, i.e., obtain all pairs (r, t), $r \in R$, $t \in T$ such that r intersects t at any time. Note that in this problem trajectories can be thought as LineStrings because the temporal dimension is not considered in the queries. A common approach to solve this kind of problem assumes a filter and refinement process. In the filter step a simplification of the spatial regions (e.g., the minimum bounding rectangles, MBRs) is used to obtain a set of candidates, which are later refined in the second step by comparing real spatial regions.
We could consider trajectories as spatial objects with similar characteristics to regions, and use their MBRs. Thus, this problem reduces to the spatial-join of MBRs, which solution in MapReduce frameworks has been studied in [CLUSTER'09]. The main problem of this approach is that MBRs of trajectories usually produce large empty regions, which yield many false positives that have to be refined, and this produces a deterioration in the efficiency of the algorithm.
Therefore, we use the same solution in [CLUSTER'09] for the set of regions R and propose new techniques to handle the set of trajectories T. Before presenting our results, let us summarize the features of MapReduce and the proposal in [CLUSTER'09]. MapReduce is a framework that simplifies distributed computing on clusters of share-nothing machines. It hides details about data distribution, availability of data, fault-tolerance, etc., so the programmer just focus on data processing. This processing consists of two stages, namely Map and Reduce, and inputs/outputs of both stages are lists of (key, value) pairs. This simple scheme is common to many parallel applications. In the map step each node processes a set of inputs and generates a set of outputs of the form (key, value). Then, all the map outputs with the same key are processed by the same reducer, which also generates (key, value) outputs. Thus, a key factor for the performance of this kind of systems is the balancedness of the of work-load of the nodes.
Intrinsically to its nature, spatial information is usually skewed and this turns balancedness in parallel computation a challenge. In [CLUSTER'09] authors propose the decomposition of the space in a set of tiles of the same size (size in this case means extension and not number of objects). Thus, each spatial object belongs to a set of tiles. The bigger the tile, the smaller the replication factor. However, big tiles do not work well with skewed data in terms of work-load balancedness. Hence, the tile size offers a trade-off. Then, tiles are coded according with a space filling curve (Z-order offers the best results in their experiments). They distribute these tiles into the different partitions using a round-robing scheme. In the reduce step, they first perform

a filter (pairing tuples according with their MBRs) using a *strip-based plane sweeping* algorithm, and then a refinement with well-known algorithms.
When working with moving objects, input files are usually a mess of positions reported by objects at specific timestamps (i.e., these data are not structured in trajectories or segments). In addition, these files are huge and any preprocessing should also been performed in a distributed environment. We observed that the problem of building trajectories from points is equivalent to the construction of inverted files, and this problem has been well studied in the MapReduce context (see for example the book of Lin and Dyer).
Once we have the trajectories, we could split and distribute them using a tiling scheme similar to CLUSTER'09. However, we propose a new tiling scheme where the size of the tile is adaptive to the distribution of the data. The idea is to keep tiles as big as possible in order to avoid unnecessary replication of data. Thus, in sparse areas the size of the tile should be larger than in dense areas. To determine the tile size, we use a MapReduce phase to build an adaptive tile-partition over a sampling of the input. The algorithm is very similar to the construction of a k-d-tree. We consider two variants of this algorithm. In the first one the sampling is performed over the raw data (points), whereas in the second one the sampling is performed over segments of the trajectories. The computational cost of the second one is higher, but it may improve the global performance as it may avoid situations where segments are systematically partitioned in several tiles.
Finally, the spatial-join is performed in another MapReduce phase. In the map step trajectories and regions are partitioned and distributed according to the tiling scheme computed in the previous phase. In the reduce step the spatial-join is actually performed using a plane-sweep algorithm. We also propose an optimized version of this algorithm for the special case when the size of regions dataset fits in main memory. Recall that we are working with the MBRs of the regions and this is a reasonable assumption for real datasets. This optimized version uses the distributable cache mechanism provided by MapReduce to make some data available to all nodes in the cluster.

4. Future collaboration with host institution (if applicable)

As we mentioned before, with this STSM we started a collaboration in a hot topic with many open problems. Our immediate collaboration will focus on finish the work that we described in the previous section and on its extension for two sets of trajectories. We also identified the following open problems (all of them in the context of MapReduce) as targets for future collaboration:

- Bulk-loading classical trajectory indexes. Even though moving objects generate massive datasets, it may be possible to use distributed computation to generate indexes that can run in a single server.
- Top-k queries on trajectory dataset. The idea of this kind of queries is to provide responses that contain just the k most relevant results (and the computational cost should be proportional to this parameter k).
- Clustering and sampling. This kind of techniques that provide a summary of the whole dataset are becoming essential in the field of big data.

5. Foreseen publications (if applicable)

We are currently implementing and evaluating the algorithms described in Section 3. We plan to complete an empirical evaluation of the ideas presented by the end of the year. Therefore, we expect to submit this work to a conference in 2013. If we get a

positive feedback, we can extend this work and submit a journal version by the end of 2013. In addition, we plan to present our results in the next MOVE-COST meeting.

References

[CloudDB'09] Qiang Ma, Bin Yang, Weining Qian, Aoying Zhou: Query processing of massive trajectory data based on mapreduce. CloudDb 2009: 9-16.

[CLUSTER'09] Shubin Zhang, Jizhong Han, Zhiyong Liu, Kai Wang, Zhiyong Xu: SJMR: Parallelizing spatial join with MapReduce on clusters. CLUSTER 2009: 1-8.

[EDBT'10] Foto N. Afrati, Jeffrey D. Ullman: Optimizing joins in a map-reduce environment. EDBT 2010: 99-110.

[SIGMOD'11] Alper Okcan, Mirek Riedewald: Processing theta-joins using MapReduce. SIGMOD Conference 2011: 949-960.

[SIGMOD'96] Jignesh M. Patel, David J. DeWitt: Partition Based Spatial-Merge Join. SIGMOD Conference 1996: 259-270.

[SSDBM'09] Ariel Cary, Zhengguo Sun, Vagelis Hristidis, Naphtali Rishe: Experiences on Processing Spatial Data with MapReduce. SSDBM 2009: 302-319.