ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-4/W2, 2015
International Workshop on Spatiotemporal Computing, 13–15 July 2015, Fairfax, Virginia, USA

A SPATIOTEMPORAL AGGREGATION QUERY METHOD USING MULTI-THREAD PARALLEL TECHNIQUE BASED ON REGIONAL DIVISION

Shuai Liao*, Luo Chen, Jun Li, Wei Xiong, Qiuyun Wu

College of Electronic Science and Engineering, National University of Defense Technology, Changsha, Hunan Province, China - liaoshuai@whu.edu.cn, (luochen, junli, weixiong, qiuyuwu)@nudt.edu.cn

KEY WORDS: Moving Objects, Spatiotemporal Aggregation, Multi-thread Parallel, Regional Division

ABSTRACT:

Existing spatiotemporal database supports spatiotemporal aggregation query over massive moving objects datasets. Due to the large amounts of data and single-thread processing method, the query speed cannot meet the application requirements. On the other hand, the query efficiency is more sensitive to spatial variation then temporal variation. In this paper, we proposed a spatiotemporal aggregation query method using multi-thread parallel technique based on regional division and implemented it on the server. Concretely, we divided the spatiotemporal domain into several spatiotemporal cubes, computed spatiotemporal aggregation on all cubes using the technique of multi-thread parallel processing, and then integrated the query results. By testing and analyzing on the real datasets, this method has improved the query speed significantly.

1. INTRODUCTION

In recent years, with the rapid development of location technology and spatiotemporal database, it becomes more convenient and efficient to implement data acquisition, storage and management of large-scale moving objects, such as the trajectory data of vehicles in traffic network and individual check-in data. Mining and analyzing these data can discover more useful information for people's production and life. Spatiotemporal aggregation query is one of the effective ways.

Spatiotemporal aggregation query is to get a certain dataset by dividing the data tuples according to different spatiotemporal granularities, then execute corresponding aggregate functions to calculate statistical information of relative attributes on this dataset. By aggregation computing on the data of large-scale moving objects, we can obtain integral statistical information and variation tendency of moving objects, such as the variation of the number of simultaneously moving cars in ROI, the speed variation of vehicles on some particular roads and the information of traffic flow in certain crossroads, to provide theoretical basis for urban construction. (Gennady and Natalia, 2010) visualized the results of spatiotemporal aggregation query over vehicles' trajectory data in the form of thematic maps and graphics. On this basis, they analyzed the continuous variation of attribute information with the distribution of space and time in the urban traffic network, such as the stream of traffic, driving speed and direction, and then monitored the condition of traffic (Natalia and Gennady, 2013).

2. RELATED WORK

SQL standard provides a range of aggregate functions. Therefore, the existing spatiotemporal database supports spatiotemporal aggregation query over large-scale data of moving objects. However, as the data volume is large, and single-thread processing is most widely used, the query efficiency cannot meet the application requirements. To solve this problem, there are generally three solutions: index-based aggregation algorithm, non-indexed aggregation algorithm and the algorithm that integrates index-based aggregation with pre-aggregation (Lopez et al., 2005).

(Papadias et al., 2001) proposed an index-based approach in which they group spatial objects into static regions and index these regions using an R-tree. Each region represented in the R-tree is annotated with aggregate information over all the timestamps in the base relation. In addition, for every region represented in the R-tree, the proposed data structure maintains a B-tree that contains time-varying aggregate data about the region. This data structure is called the aggregation RB-tree (aRB-tree) and can be extended to handle the case when objects are grouped into dynamic regions, resulting in the aggregation historical RB-tree or the aggregation three-dimensional R-B-tree (Tao and Papadias, 2005). One drawback of Tao et al.'s aRB-tree is the distinct counting problem. (Tao et al., 2004) recognized the distinct counting problem within the aRB-tree and presented an approximate approach to evaluate distinct COUNT and distinct SUM aggregate queries. Their approach is based on sketches and a sketch index, and it is similar with the aRB-tree in structure. (Lee et al., 2006 and 2009) focused on efficient index structure for spatiotemporal aggregation of trajectory in a constrained network, named aCN-RB-tree.

Another approach for the approximate evaluation of spatiotemporal aggregation query was proposed by (Sun et al., 2004). Space is partitioned in a two-dimensional grid of w×w regular cells, where w is a constant called resolution. Each cell is associated with the number of objects within it. Sun et al.'s approach can answer approximate queries to the COUNT aggregate based on a data structure termed Adaptive Multidimensional Histogram (AMH), which is updated every time an object transmits a new position. The AMH can only be used for answering snapshots aggregate queries. (Braz et al., 2007) discussed how data warehousing technology can be used to store aggregate information about trajectories and perform OLAP operations over them.

Tao and Papadias integrated spatiotemporal indexes with pre-aggregation, and proposed a method supporting dynamic spatiotemporal dimensions for the efficient processing of

* Corresponding author

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
doi:10.5194/isprsannals-II-4-W2-1-2015

historical aggregation query without a priori knowledge of grouping hierarchies (Tao and Papadias, 2005).

In this paper, for the historical trajectory data of moving objects, we used COUNT and GROUP BY functions to compute spatiotemporal aggregation, and proposed a spatiotemporal aggregation query method using multi-thread parallel technique based on regional division. Specifically, we divided the spatiotemporal domain into several spatiotemporal cubes on the server, and computed spatiotemporal aggregation on all cubes using the method of multi-thread parallel technique. Then, we integrated all of the query results. This method was built on a structure of multiple indexes in the original moving objects database (MODB). Compared with the traditional spatiotemporal aggregation query, the computational efficiency of this method has improved significantly.

3. MULTI-THREAD PARALLEL ALGORITHM

3.1 Approach to Divide Spatiotemporal Cube

A spatiotemporal cube is defined by users, as shown in Figure 1. Users can specify the spatial region and temporal range according to their interests, which can be abbreviated to ROI and TOI. The temporal granularity Δt which is the basis for GROUP BY also needs to be specified and the value should meet the application requirements. According to the user-specified parameters, the algorithm queries the set of moving objects within the spatiotemporal region of users' interests in database. Meanwhile, it divides TOI into several time segments according to Δt, and groups the set depending on the time segments of TOI, gets the moving objects within each time segment in ROI, and then calculates the spatiotemporal aggregation by COUNT aggregate function.

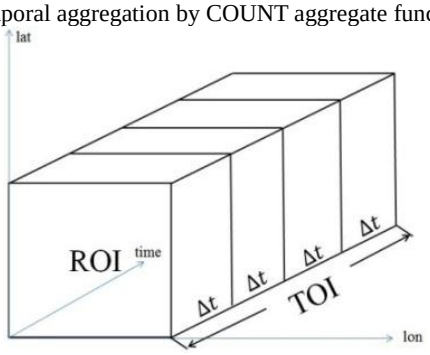
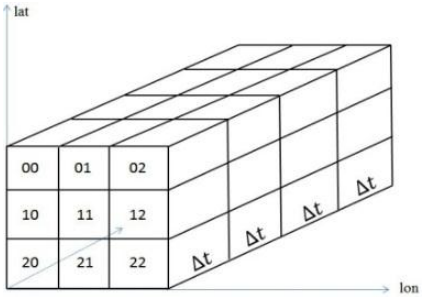


Figure 1. The user-defined ST-Cube

Due to the feasibility of parallelism on spatiotemporal aggregation and the conclusion of experiment 1 in Section 4: the effect of spatial variation on spatiotemporal aggregation query efficiency is significantly greater than that of temporal variation. We divided the spatial region ROI evenly into a series of subspace according to a spatial granularity N. As shown in Figure 2, we divided ROI evenly into 9 spatiotemporal sub-cubes with 3×3 cells.



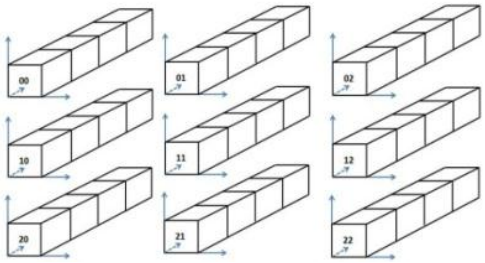


Figure 2. Diving ROI evenly into 3×3 cells and generating 9 spatiotemporal sub-cubes

Meanwhile, according to multi-thread parallel programming, the amount of threads is equal to the number of spatiotemporal sub-cubes, so that each sub-thread corresponds to a spatiotemporal sub-cube. After all sub-threads get the spatiotemporal aggregation parameters, they will generate SQL query statements that MODB can identify and execute them to get the results of spatiotemporal aggregation in parallel.

This algorithm employs a parallel approach of task division essentially.

3.2 Flow of Multi-thread Parallel Algorithm

Figure 3 shows the flow of multi-thread parallel algorithm of spatiotemporal aggregation query.

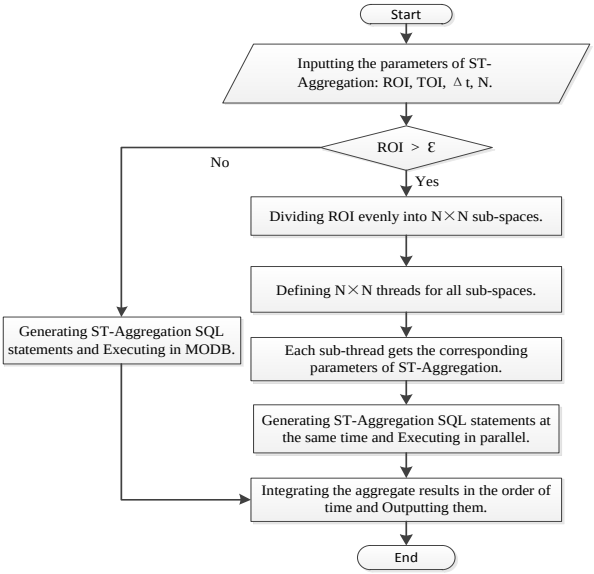


Figure 3. Parallel algorithm flow of spatiotemporal aggregation query over moving objects

4. THE IMPLEMENTATION OF ALGORITHM AND ARCHITECTURE OF SPATIOTEMPORAL AGGREGATION QUERY SERVICE

4.1 Implementation of Algorithm

This algorithm is concretely implemented by several sub-modules on the server, including sub-module of regional division, sub-module of multi-thread task allocation, sub-module of aggregate query parsing and sub-module of aggregate results integrating.

Sub-module of regional division compares ROI with a given

spatial threshold Ɛ. If ROI is larger than Ɛ, it will divide the spatial region ROI evenly. The function of this module is to provide the subspace (represented by Sub_ROI) for sub-module of multi-thread task allocation. According to the experiments, we know that there's no need to use the approach of multi-thread parallel processing if ROI is too small and the efficiency of aggregate calculation could meet the need of users. Hence, we set a spatial threshold Ɛ and its value is gained based on experience.

Sub-module of multi-thread task allocation defines Threading class firstly, and instantiates Threading class according to the number of Sub_ROI. Then it allocates a Sub_ROI which need to calculate spatiotemporal aggregation to each sub-thread by means of multi-thread parallel.

Sub-module of aggregate query parsing uses every sub-thread to parse the parameters into SQL query statement that MODB can identify, and then calculates spatiotemporal aggregation in database. The example of SQL query statement is as follows:

```
SELECT Moving_object_time, COUNT (DISTINCT
Moving_object_id)
FROM Moving_object_table
WHERE INTERSECTS (Moving_object_position, Sub_ROI)
AND Moving_object_time IN TOI
GROUP BY Δt
ORDER BY Moving_object_time
```

Sub-module of aggregate results integrating obtains spatiotemporal aggregate results of every sub-thread, and integrates them in order of time. Then it sends the integrated results to the client via HTTP service.

4.2 Architecture of Spatiotemporal Aggregation Query Service

By implementing this algorithm, we encapsulated it as spatiotemporal aggregation query service on the server. Here is the simple architecture shown in Figure 4.

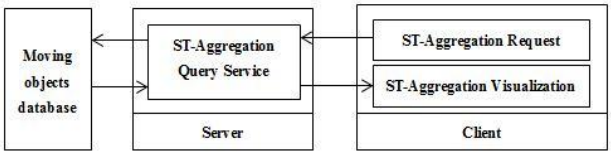


Figure 4. Architecture of aggregation query service

The client sends a query request and delivers the user-defined parameters obtained from front end to spatiotemporal aggregation query service on the server. And these user-defined parameters consist of the spatial region that user interests ROI, the temporal range that user specifies TOI, the time granularity Δt and the spatial granularity N. In addition, the client receives the aggregate results which are provided by spatiotemporal aggregation query service, and visualizes them based on the open source library of Data-Driven Documents (D3.js) on the web browser.

As shown in Figure 5, (a) is the statistical graph of the amount of taxis in Beijing during one week for each hour, (b) is the statistical graph of the amount of sample points of taxis in Beijing during one week for each hour, (c) is the statistical graph of the amount of taxis on Changan Avenue during one week for each hour, (d) is the statistical graph of the amount of sample points of taxis on Changan Avenue during one week for each hour.

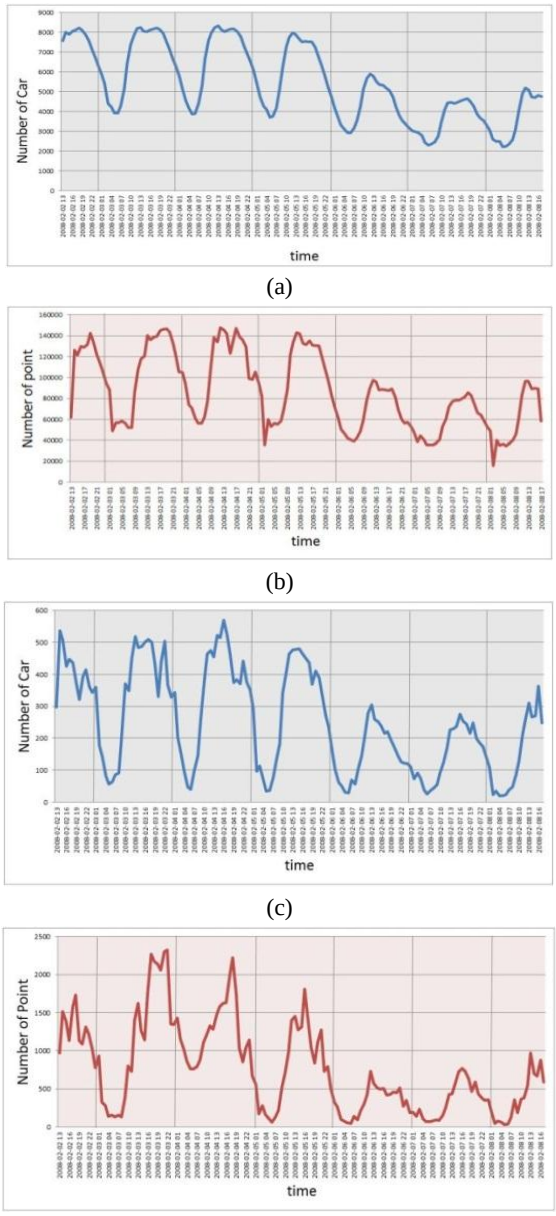


Figure 5. The visualization results of spatiotemporal aggregation query

The MODB is based on the open source Object-Oriented Relation Database (ORDB) PostgreSQL. We extend it in spatial attribute, and store the historical trajectory data of moving objects in the form of point model. Multiple indexes are created on this data model, gist index of PostgreSQL on the spatial domain, and B-tree index on the temporal domain. The traditional spatiotemporal aggregation queries are built on this structure of multiple indexes (represented by Indexes in section 5).

On the other hand, the moving objects will be counted more than once that fall on the grid lines when we divide the spatial region. This will result in some deviations compared with the actual situation. In order to estimate the validity of aggregate results, we define a concept of Error Threshold. If the Average Error Ratio is smaller than Error Threshold, the aggregate results should be valid. Below is the formula of Average Error Ratio (AER), as shown in Formula (1).

$$\mathrm{AER} = \frac{1}{n}\sum_{1}^{n}\frac{Result(multi\_thread)_i - Result(Indexes)_i}{Result(Indexex)_i} \quad (1)$$

where $n = \frac{TOI}{\Delta t}$ and i means the time segment in n.

5. EXPERIMENT RESULTS AND ANALYSIS

In order to verify the validity and feasibility of this parallel algorithm, we tested it with two experiments on real datasets, and compared the efficiency of spatiotemporal aggregation query based on the structure of multiple indexes with that of this algorithm. In the process of experiments, we specified the time granularity Δt to be one hour and assigned AER 0.1%.

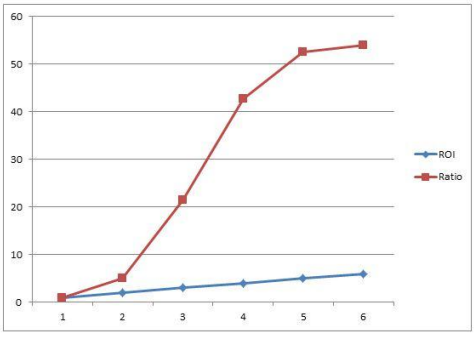
Experimental datasets: dataset 1 is the historical trajectory data of more than 10,000 taxis in Beijing from February 2, 2008 to February 8, 2008, and includes more than 15,000,000 GPS points with timestamps (Yuan et al., 2011); dataset 2 is the GPS historical trajectory data of 182 volunteers who collected from August, 2007 to August, 2012 and includes more than 25,000,000 GPS points with timestamps, and most of them are located in Beijing (Zheng et al., 2009).

Experimental environment: Intel Core i5-2300 2.8GHz 4 CPUs; 4.00GB RAM; RHEL 6.3 OS; Eclipse IDE; ORDB PostgreSQL.

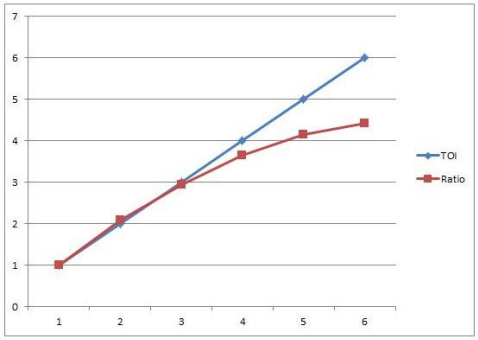
Experiment 1: we calculated spatiotemporal aggregation based on the structure of multiple indexes on dataset 1, and verified the effect of spatial variation and temporal variation on the efficiency of spatiotemporal aggregation query. Figure 6 shows the corresponding results. Formula (2) shows the value of Ratio in Figure 6.

$$Ratio_i = \frac{Execution\_Time_i}{Execution\_Time_1} \quad (2)$$

where i means the time of experiment.



(a)



(b)

Figure 6. The effect of ROI and TOI variation on spatiotemporal aggregation query efficiency

From (a), we know that when ROI grew linearly, the expense of spatiotemporal aggregation query increased significantly; but from (b) we can see that when TOI grew linearly, the variation tendency of expense of spatiotemporal aggregation query was not obvious. Therefore, we conclude that the effect of spatial variation on spatiotemporal aggregation query efficiency is significantly greater than that of temporal variation. And this is the reason that we divided the spatial region ROI evenly using the method of multi-thread parallel technique when calculated spatiotemporal aggregation.

Experiment 2: first, we compared the efficiency of spatiotemporal aggregation query on dataset 1 based on the structure of multiple indexes with that of multi-thread parallel technique, we only used 9 threads in this experiment. As shown in Figure 7, the horizontal axis represents the relative size of spatial region ROI, and the vertical axis represents the execution time of spatiotemporal aggregation query. We can see from the graph that the efficiency of spatiotemporal aggregation based on multi-thread parallel technique increased significantly compared to the structure of multiple indexes.

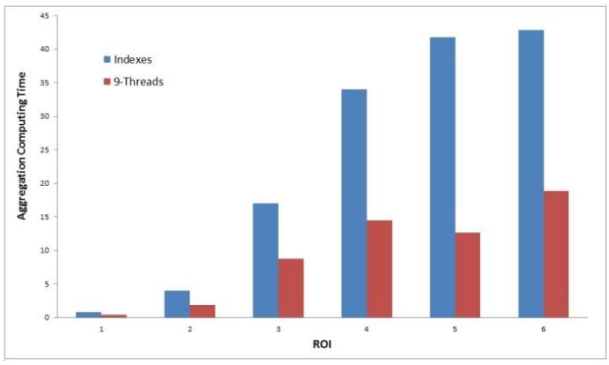
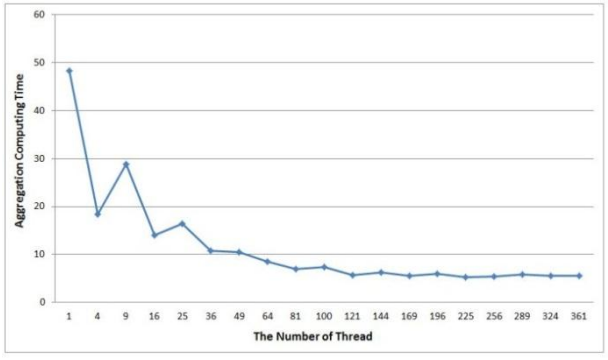
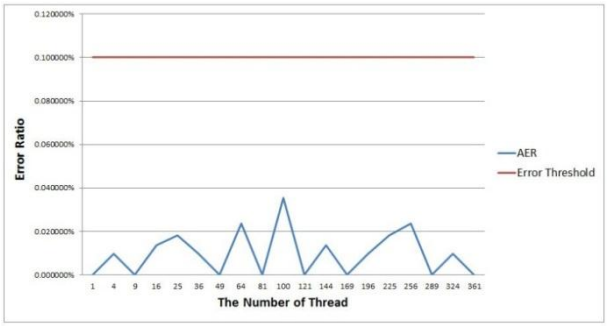


Figure 7.Comparison of spatiotemporal aggregation query efficiency between the structure of multiple indexes and multi-thread parallel technique

Then we verified the variation tendency of query efficiency and AER with the increased amount of threads. Figure 8 shows the results. From (a), we know that the efficiency of spatiotemporal aggregation query improved continually with the increased number of threads. When the number of threads reached a certain threshold, the efficiency tended toward stability. In general, the efficiency of spatiotemporal aggregation query based on multi-thread parallel technique has been improved by about 9 times on dataset 1. From (b), we can see that the data error was inevitable when divided the spatial region ROI evenly. But the value of AER varied randomly and would not exceed the Error Threshold 0.1%.



(a)



(b)

Figure 8. The efficiency of multi-thread parallel method and the variation of AER

Finally, in order to further prove the validity of this parallel approach, we tested it on dataset 2 with the same experiments. As shown in Figure 9, the efficiency of spatiotemporal aggregation query based on this approach has been improved by about 5 times on dataset 2. The approach of multi-thread parallel will certainly improve the efficiency of spatiotemporal aggregation query on different datasets, but have different effect. This is related to the characteristics of the datasets themselves.

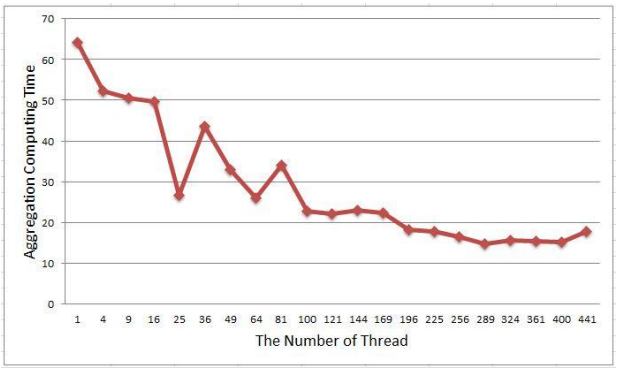


Figure 9. The experimental result of multi-thread parallel technique on dataset 2

6. CONCLUSION

In the age of big data, it has become a research hotspot in the field of spatiotemporal database that storing, managing, mining and analyzing large-scale data of different kinds of moving objects. Spatiotemporal aggregate calculation is one of the important aspects. By spatiotemporal aggregate calculation, we can obtain integral statistical information and variation tendency of moving objects which has great significance and application value in real life and production. Therefore, it becomes particularly important to improve the efficiency of spatiotemporal aggregate calculation.

This paper summarized the related work on the aspect of spatiotemporal aggregation, and proposed a spatiotemporal aggregation query method using multi-thread parallel technique based on regional division. By testing and analyzing on the real datasets, this method has improved the query speed significantly compared with the traditional spatiotemporal aggregation query based on the structure of multiple indexes.

What our experiments further discovered was that the computational efficiency of this approach was limited by the spatiotemporal sub-cube which has the maximum amount of data. Therefore, we can consider designing an unevenly division approach based on heuristic rules through generating the histogram of spatiotemporal data, to balance the computation of each sub-thread, and then to further improve the computational efficiency.

REFERENCES

Braz, F. et al., 2007. Approximate Aggregation Trajectory Data Warehouse. ICDE 2007, pp. 536-545.

Gennady, A. and Natalia, A., 2010. A General Framework for Using Aggregation in Visual Exploration of Movement Data. The Cartographic Journal, 47(1), pp. 22-40.

Lopez, I. et al., 2005. Spatiotemporal Aggregation Computation: a Survey. IEEE TKDE, 17(2), pp. 271-286.

Lee, D.W. et al., 2006. Constraint Network Based Index for Spatio-Temporal Aggregation of Trajectory in Spatial Data Warehouse. Journal of Korea Multimedia Society, 9(12), pp. 25-37.

Lee, D.W. et al., 2009. aCN-RB-Tree: Update Method for Spatio-Temporal Aggregation of Moving Object Trajectory in Ubiquitous Environment. International Conference on Computational Science and Its Applications, pp. 177-182.

Natalia, A. and Gennady, A., 2013. Visual Analytics of Movement: an Overview of Methods, Tools, and Procedures. Inf Vis, 12(1), pp. 3-24.

Papadias, D. et al., 2001. Indexing Spatio-Temporal Data Warehouses.Proceedings Of The 20th International Conference On Data Engineering, pp. 166-175.

Sun, J. et al., 2004. Querying about the Past, the Present, and the Future in Spatio-Temporal Databases. ICDE 2004, pp. 202-213.

Tao, Y. et al., 2004. Spatio-temporal Aggregation Using Sketches. ICDE 2004, pp. 214-225.

Tao, Y. and Papadias, D., 2005. Historical Spatio-Temporal Aggregation. ACM TOIS, 23(1), pp. 61-102.

Yuan, J. et al., 2011. Driving with Knowledge from the Physical Word. In Proc.KDD 2011, ACM Press 2011.

Zheng, Y. et al., 2009. Mining Interesting Locations and Travel Sequences from GPS Trajectory for Mobile Users. Proceedings of the 18th international conference on World Wide Web, ACM, pp. 791-800.