FACTA UNIVERSITATIS
Series: **Automatic Control and Robotics** Vol. 14, N° 1, 2015, pp. 19 - 27

PROCESSING AND ANALYSIS OF BIG TRAJECTORY DATA USING MAPREDUCE

UDC 004.62/.65:622.716

Natalija M. Stojanović[1], Dragan H. Stojanović[1]

[1]University of Niš, Faculty of Electronic Engineering, Department of Computer Science, Niš, Republic of Serbia

Abstract. *In this paper, we present research work related to processing and analysis of big trajectory data using MapReduce framework. We describe the MapReduce-based algorithms and applications implemented on Hadoop for processing spatial join between big trajectory data and set of POI regions and appropriate aggregation of join results. The experimental evaluation and results in detecting trajectory patterns of particular users and the most popular places in the city demonstrate the feasibility of our approach. The visual analytics of MapReduce job output improve the trajectory and movement analysis.*

Key words: *data processing, geospatial analysis, high performance computing, parallel processing, spatial databases*

1. INTRODUCTION

Advances in remote sensors, sensor networks, and the proliferation of location sensing devices lead to the massive generation of dynamic and geographically distributed spatio-temporal data in the form of moving object trajectories. These ever-increasing volumes of spatio-temporal data call for new models and computationally effective algorithms for efficient storage, processing, analysing and visualization in advanced data-intensive systems and applications.

During the last decade there has been a growing interest in research of parallel and distributed computing applied to the management, processing and analysis of massive geo-spatial data [1]. Advanced GIS applications, such as emergency management, climate change analysis, traffic monitoring, smart cities, etc., impose strengthen performance and response time constraints which cannot be met by contemporary Geographic Information

Received December 11, 2014
Corresponding author: Natalija M. Stojanović
University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Republic of Serbia
E-mail: natalija.stojanovic@elfak.ni.ac.rs

Systems (GIS) and spatial databases. Thus, high-performance computing (HPC) may meet the requirements of these applications [2].

Various high performance methods and techniques have been proposed for processing and analysis of big spatial data based on cluster and cloud computing [3], as well as on personal computers equipped with multiprocessor CPUs and massively parallel GPUs [4]. The recent proliferation of distributed and cloud computing infrastructures, both public clouds (e.g., Amazon EC2) and private computer clusters, has given a rise for processing and analysis of complex Big data. Especially, the implementation of MapReduce framework as open-source Hadoop software stack, that can work on clusters of commodity computers, have set this paradigm as an emerging research and development topic [5]. The MapReduce paradigm hides details about data distribution, data availability and fault-tolerance, and can scale to thousands of computers in a cluster or cloud. The MapReduce processing consists of two phases, namely Map and Reduce that are performed through map and reduce functions over data records formatted as key-value pairs [6].

In this paper we implement two MapReduce jobs and corresponding applications using Hadoop to perform spatial join between trajectory data set and places of interest (POI), and further aggregation of join results, to generate the symbolic trajectories of mobile users, as well as to detect the most popular POI in the city.

The rest of the paper is structured as follows. Section II presents the research work related to processing and analysis of spatial and spatio-temporal data using MapReduce. In section III we describe the Hadoop implementations for processing of big trajectory data set over set of places of interest (POI). Section IV gives the results and presents the evaluation of our implementation. Section V presents the visual analysis of the results. Section VI concludes the paper and gives directions for future research.

2. RELATED WORK

Recently, there is a growing research interest in spatio-temporal data management, processing analysis and mining using MapReduce model [7].

Cary et al. in [8] present their experiences in applying the MapReduce framework to important spatial database problems. They investigate R-tree bulk-loading issues in MapReduce, as well as aerial image quality computation and prove excellent scalability in parallel processing of spatial data. In [9] some efficiency issues regarding spatial data management are considered and an implementation of the all-nearest-neighbor query algorithm is provided. The authors present performance evaluation and show that the MapReduce-based spatial applications outperform the traditional one on a DBMS.

Spatial joins in MapReduce are studied in [10]. The authors present SJMR (Spatial Join with MapReduce) algorithm that includes strip-based plane sweeping algorithm, tile-based spatial partitioning function and duplication avoidance technology to perform spatial join on MapReduce. The performance evaluation of SJMR algorithm over the real-world data sets shows the applicability of MapReduce for data-intensive spatial applications on small clusters.

Regarding spatio-temporal and trajectory data, a first approach is presented in [11] where massive trajectory management issues are investigated. The authors present a new framework for query processing over trajectory data based on MapReduce in order to

utilize the parallel processing power of computer clusters. They perform preliminary experiments showing that this framework scales well in terms of the size of trajectory data set [12].

SpatialHadoop is developed as the first extension of MapReduce framework with support for spatial data and operations [13]. SpatialHadoop employs a high level spatial language, a two-level spatial index structure, and three basic spatial operations: range queries, k-NN queries, and spatial join. SpatialHadoop demonstration has been done on an Amazon EC2 cluster against two real spatial data sets.

Although there is a considerable recent research interest related to spatial and spatio-temporal data management on MapReduce, there is limited work performed for trajectory (mobility) data processing and analysis using the MapReduce framework. Our work aims to provide efficient MapReduce solution for a fundamental mobility data processing task related to big trajectory data sets.

3. Trajectory Data Processing and Analysis Using Hadoop

The research presented in this paper aims to provide efficient MapReduce solution for a big mobility data processing and analysis task related to the trajectory data set representing movement of mobile users and the points/places of interest they visit.

The problem we investigate in this work is actually the spatial join between a big set of spatio-temporal trajectory data T and a (potentially large) set of spatial regions R. The trajectory data represent the movement of a large collection of moving objects/mobile users tracked for a certain time period with the specified frequency of location updates. Each trajectory is seen as a collection of points $<oid, x_i, y_i, t_i>$, where x_i, y_i represent the location in a geographic/geometric reference system and t_i is the corresponding time stamp at which the moving object (*oid*) is detected at the specified location. Trajectories of a large number of moving objects collected continuously for a long time period are characterized by very large volumes, considered as Big Data and therefore their processing and analysis is a challenging issue.

Each spatial region R represents an area around point/place of interest (POI) visited by mobile users that stay there for certain time periods. A mobile user visits particular POI if its recording location at corresponding time stamp is within the area of POI; otherwise a mobile user is considered to be on a trip between two POIs.

The objective of our MapReduce implementation is to provide:

- Analysis of user's movement and trajectory to detect places she visited and at which she stayed for a certain time period in the form of symbolic trajectory $(POI_1,Period_1)\rightarrow(POI_2,Period_2)\rightarrow\ldots\rightarrow(POI_n, Period_n)$.
- Detection of the most popular places regarding the number of users that visited them and the total amount of time they stayed at a particular place.

For such analysis we develop two MapReduce application/jobs over the trajectory and region data sets.

The first job, named *SemanticTrajectory*, performs processing and analysis of trajectory data related to mobile users and detects their visits to POIs (Fig. 1). During the Map phase, each mapper reads its input, which is a collection of records of the form <*oid*,

location, time> and performs the spatial join with POI data set containing records of the form <*pid, area, attributes*>, according to the spatial relation Within(location, area). The output of mappers is in the form <(*oid, pid*), *time*> where the pair (*oid, pid*) represents the composite key and the parameter time is a value. In the Reduce phase, each reducer collects the identifiers of the same (*oid, pid*) and process and aggregate the time values detecting the time period(s) during which the object stays at the POI. The output of the reducers contains records of the form <*oid, pid,* (t_1,t_2)> and is written back to HDFS. Each record of the output represents the period during which a mobile user *oid* visits the place *pid* and represents the semantic trajectory of a mobile user, i.e. the symbolic pattern of user's movement.

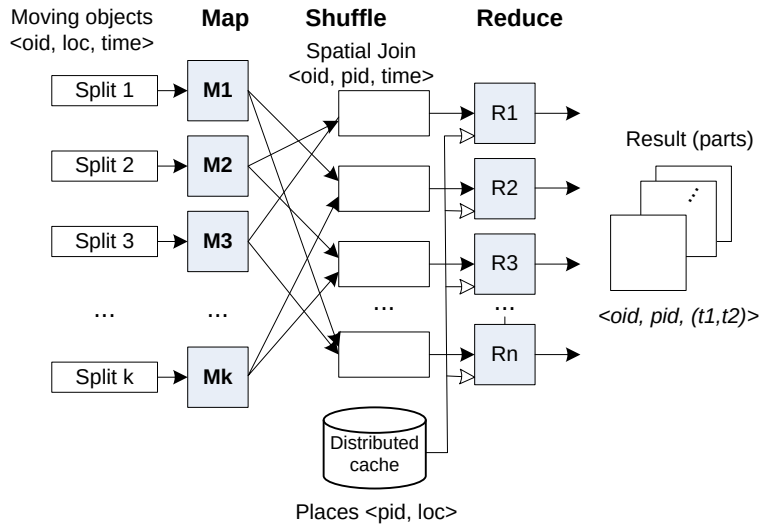

Fig. 1 The outline of the *SemanticTrajectory* job

The second MapReduce job we developed, named *PopularPlaces*, focuses on POIs and detects their popularity according to the number of visits and the total duration of stays. During the Map phase, each mapper reads the same big trajectory data set as the first job and performs the spatial join with POI data set. This time the output of mappers is in the form <*pid, (oid, time*)> where the pid represents the key and (*oid, time*) is a value. In the Reduce phase, each reducer collects the identifiers of the same *pid*, and process and aggregate the *oid* and *time* values detecting the total number of unique mobile users that visited particular place and the total time periods of their visits to POI. The output of the reducers contains records of the form <*pid, nr_oid, total_time*> and is written back to HDFS. Each record of the output represents the total number of users (*nr_oid*) that visited a place *pid*, and the total time (*total_time*) that these users stay at place *pid*.

4. EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of the implemented algorithms described previously.

The algorithms have been implemented in Hadoop 1.2.1 (http://hadoop.apache.org/) and experiments have been conducted in a pseudo-distributed mode, as well as on a small cluster of 5 nodes/commodity computers. A master node is a physical machine Pentium IV with 3 GHz CPU and 4GB of RAM, while worker nodes are virtual machines on a private IaaS cloud equipped with Intel Xeon CPU 1.6GHz Dual Core. Each node runs Ubuntu 12.10 Linux and has Java SDK 1.7 installed. In addition, each node runs both a Task Tracker and Data Node daemon, while a master node acts as Job Tracker and Name Node, as well.

In implementation and evaluation of our algorithms we have used MilanoByNight simulated datasets that have been provided by the EveryWare Lab, University of Milano [14]. The authors of the data set consider a typical deployment scenario for a friend-finder service: a large number of young people using the service on a weekend night in large city like Milan. The simulation includes a total of 30,000 home buildings, 10,000 office buildings and 1,000 entertainment places which represent a POI dataset. The trajectory data set contains 180 million of records for 100,000 mobile users moving over the city of Milan while location updates are made at every 2 minutes. The movement has been recorded over 6 hours long time period, from 7pm - 1 am, which amounts for about 1.05 GB in total.

Both data sets are stored in the HDFS. The trajectory data set is available to all started mappers through HDFS partitioning mechanism. For the POI data set, we exploit the features of the distributed cache mechanism supported by Hadoop, meaning that all POI data are available to all Map and Reduce tasks. Since in our setting the size of the POI dataset is significantly smaller than the size of the trajectories dataset, the distributed cache is a convenient way to share data across Hadoop nodes.

We have run our MapReduce jobs on a small commodity cluster containing 5 nodes. The *SemanticTrajectory* MapReduce job engages 17 Map tasks and 10 reduce tasks that finish the job in 8 minutes and 15 seconds in average, producing an output of about 56 MB stored on HDFS. The job output is in the form of records shown on Table 1 for particular users.

Table 1 The output of the *SemanticTrajectory* job.

OID	PID	Time period
10233	2091	[09.01.2009. 07:02 - 09.01.2009. 07:52]
10233	1362	[09.01.2009. 08:58 - 09.01.2009. 11:30]
10233	4560	[09.01.2009. 11:45 - 10.01.2009. 00:58]
...		
14215	3195	[09.01.2009. 07:00 - 09.01.2009. 08:30]
14215	1587	[09.01.2009. 08:42 - 09.01.2009. 10:04]
14215	1890	[09.01.2009. 10:24 - 09.01.2009. 12:02]
14215	2964	[09.01.2009. 12:10 - 10.01.2009. 1:00]
...		

The *PopularPlaces* job engages the same number of Map and Reduce tasks and performs faster than previous one, completing its processing for 7 minutes and 35 second. The excerpt of the output of the *PopularPlaces* job for the most popular places, having about 100 KB in size, is shown in Table 2 sorted according the total number of visitors.

Table 2 Top popular places sorted by total number of unique visitors.

PID	Total visitors	Total time
17	1635	140892
83	1530	117016
136	1466	126504
96	1458	121378
180	1435	116852
416	1341	113436
…		

The main objective of our work is not to evaluate the overall performance of trajectory data processing, since we implement our algorithms on a small, commodity cluster. We just examine the benefits of using MapReduce paradigm in processing and analysis of big trajectory data. As an illustration, the *SemanticTrajectory* algorithm took about 21 minutes and 36 seconds processed on a computer with Intel i5 CPU/ 4GB RAM. We evaluate and fine tune the performance of *SemanticTrajectory* job by varying different Hadoop runtime parameters, such as the number of reduce tasks. The runtime is reduced by increasing the number of reduce tasks, up to the certain threshold, which is in our cluster is achieved for 10 reducers and then start to increase slowly (Fig. 2).

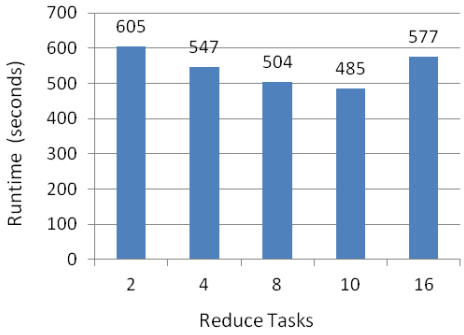

Fig. 2 The runtime of *SemanticTrajectory* job vs. number of reduce tasks

MapReduce and its Hadoop implementation provide excellent scalability in terms of bigger data sets, as well as larger computing resources. We evaluate and prove such scalability regarding increasing trajectory data set (Fig. 3).

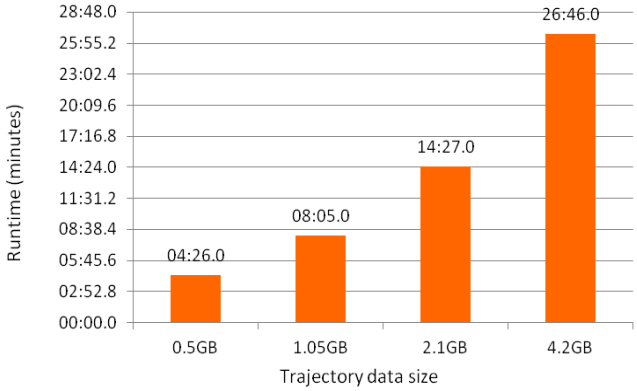

Fig. 3 The runtime of *SemanticTrajectory* job vs. trajectory data size

Also our Hadoop applications can be easily scaled to more powerful computer nodes than those we have used and much larger cluster with thousands of machines (e.g. Amazon Elactic MapReduce – EMR) to achieve much higher performances.

5. Visual Analysis of MapReduce Job Results

The results produced by *SemanticTrajectory* and *PopularPlaces* jobs can be visually analyzed using specified tools that supports analytics of spatio-temporal and trajectory data, such as Microsoft SQL Server Business Intelligence tools. The Fig. 4 shows the visual analysis of *SemanticTrajectory* job result displaying the places visited by a particular user (UserID=60127) and the time periods spent at these places.

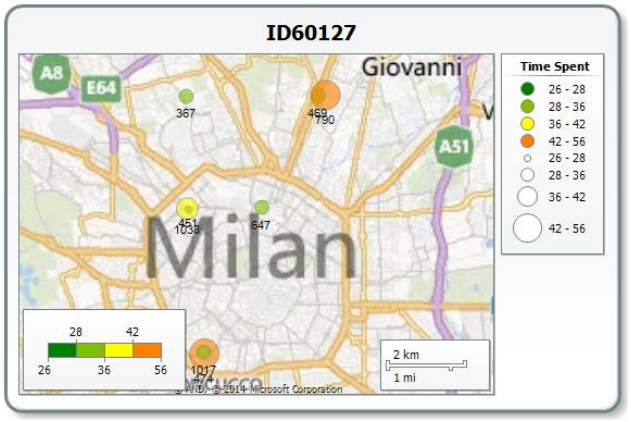

Fig. 4 The *SemanticTrajectory* job output - the places visited by user ID 60127

The Fig. 5 shows the fifty most popular places according to a number of unique visitors that visit them.

Fig. 5 The *PopularPlaces* job output – the 50 most popular places

It confirms the huge potential of visual analytics based on big trajectory data processing for discovering mobility knowledge, patterns and trends.

6. CONCLUSION

In this paper, we propose the design and implementation of efficient algorithms for processing of spatio-temporal data that represent moving object trajectories using MapReduce. These algorithms consist of spatial join between a big trajectory data set and a POI (region) data set, and appropriate aggregation of join results. The algorithms implementation has been performed using Hadoop, an open source MapReduce implementation and an Apache project. The deployment and evaluation of our solution performed on a small cluster of commodity computers, show feasibility of our approach in usability of MapReduce/Hadoop in Big spatio-temporal and trajectory data processing and analysis.

There are significant issues that are considered very important for further research and development. Since the MapReduce model is mainly batch oriented it should be interesting to explore its possibilities and extensions toward real-time stream data processing of trajectory data, as well as integration of spatio-temporal data, operation and indexing methods in Hadoop, in accordance with [13]. Since trajectories change frequently by location updates and addition of new location data, it is very interesting to explore update and monitoring issues in a MapReduce setting. Also, a very challenging task is to explore and adapt the MapReduce framework in a mobile environment (mobile cloud computing).

Acknowledgement: *This paper was realized as a part of the projects "Studying climate change and its influence on the environment: impacts, adaptation and mitigation" (III 43007), and "The infrastructure for electronically supported learning in Serbia" (III-47003), financed by the Ministry of Education, Science and Technological Development of the Republic of Serbia within the framework of integrated and interdisciplinary research for the period 2011-2014.*

REFERENCES

[1] A. Clematis, M. Mineter, R. Marciano, "High performance computing with geographical data," *Parallel Computing*, vol. 29, no. 10, pp. 1275–1279, 2003.

[2] S. Shekhar, "High performance computing with spatial datasets," in *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems - HPDGIS*, 2010, pp. 1–2. [Online]. Available: http://dx.doi.org/10.1145/1869692.1869693

[3] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, J. Saltz, "Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce," in *Proceedings VLDB Endowment*, vol. 6, no. 11, Aug. 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3814183/

[4] J. Zhang, "Towards Personal High-Performance Geospatial Computing (HPC-G)," in *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems - HPDGIS*, 2010, pp. 3–10. [Online]. Available: http://dx.doi.org/ 10.1145/1869692.1869694

[5] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of ACM*, vol. 51, no. 1, p. 107, Jan. 2008. [Online]. Available: http://dx.doi.org/ 10.1145/1327452.1327492

[6] T. White, *Hadoop: The Definitive Guide, 3rd Edition*. O'Reilly Media, 2012.

[7] V. Mayer-Schönberger, K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Eamon Dolan/Houghton Mifflin Harcourt, 2013, p. 256.

[8] A. Cary, Z. Sun, V. Hristidis, N. Rishe, "Experiences on Processing Spatial Data with Using MapReduce in Practice," in *Proceedings of 21st International Conference on Scientific and Statistical Database Management*, 2009, pp. 302 – 319. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02279-1_24

[9] K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, X. Song, "Accelerating Spatial Data Processing with MapReduce," in *Proceedings of the 2010 IEEE 16th International Conference on Parallel and Distributed Systems*, 2010, pp. 229–236. [Online]. Available: http://dx.doi.org/10.1109/ICPADS.2010.76

[10] S. Zhang, J. Han, Z. Liu, K. Wang, Z. Xu, "SJMR: Parallelizing spatial join with MapReduce on clusters," in *Proceedings of the IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/CLUSTR.2009.5289178

[11] Q. Ma, B. Yang, W. Qian, A. Zhou, "Query Processing of Massive Trajectory Data based on MapReduce," in *Proceeding of the Ffirst international workshop on Cloud Data Management - CloudDB '09*, 2009, pp. 9–16. [Online]. Available: http://dx.doi.org/10.1145/1651263.1651266

[12] B. Yang, Q. Ma, W. Qian, A. Zhou, "Truster: Trajectory data processing on clusters," in *Proceedings of 14th International Conf. on Database Systems for Advanced Applications*, 2009, pp. 768–771. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00887-0_69

[13] A. Eldawy, M. F. Mokbel, "A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data," *Proceedings VLDB Endowment*, vol. 6, no. 12, pp. 1230–1233, Aug. 2013. [Online]. Available: http://dx.doi.org/10.14778/2536274.2536283

[14] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, S. Jajodia, "On the Impact of User Movement Simulations in the Evaluation of LBS Privacy-Preserving Techniques," in *Proceedings of the 1st Internationl Workshop on Privacy in Location-Based Applications*, 2008, vol. 397. [Online]. Available: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-397/paper5.pdf