Processing of Big Spatio-Temporal Data using MapReduce

Dragan Stojanović[1], Natalija Stojanović[2]

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 aggregation of join results for the purpose of movement analysis. The experimental evaluation and results in detecting trajectory patterns of particular users and the most popular places in the city during evening demonstrate the feasibility of our approach.**

Keywords – **Spatio-temporal data processing, Big data, GIS, cloud computing, MapReduce, Hadoop**

I. INTRODUCTION

Advances in remote sensors, sensor networks, and the proliferation of location sensing devices in daily life activities lead to the explosion of disparate, dynamic, and geographically distributed spatio-temporal data in the form of moving object trajectories. Also, ground, air- and space-borne remote sensing technologies, as well as large-scale scientific simulations are generating petabytes of spatio-temporal data. These ever-increasing volumes of spatio-temporal data call for new models and computationally effective algorithms in order to efficiently store, process, analyze and visualize such a big data 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 real-time disaster management, high-fidelity terrain visualization, global climate change analysis, traffic monitoring, etc., impose strengthen performance and response time constraints which cannot be met by contemporary Geographic Information Systems (GIS) and spatial databases. Thus, high-performance computing (HPC) may meet the requirements of these applications [2]. Various researchers propose methods and techniques for high performance and parallelization in the processing and analysis of big geo-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 and platforms, 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 in the form of open-source Hadoop software stack, that can work on clusters of share-nothing machines, 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 [6]. The MapReduce processing consists of two steps, namely Map and Reduce that are performed through *map* and *reduce* functions (Fig. 1). The *map* function receives a collection of key-value pairs of the form $<k_0,v_0>$ and produces collection of key-value pairs of the form $<k_1,v_1>$, $<k_2,v_2>$, $<k_3,v_3>$, ... Then, the framework executes a *shuffle* that sends each reducer a collection $<k_i, v_1, v_2, v_3,...>$, i.e., a sequence of values corresponding to the same key value k_i. Each reducer generates its own output. Load balancing, data distribution and fault tolerance issues are handled by the MapReduce framework itself, thus, the programmer can focus on the solution to the problem.

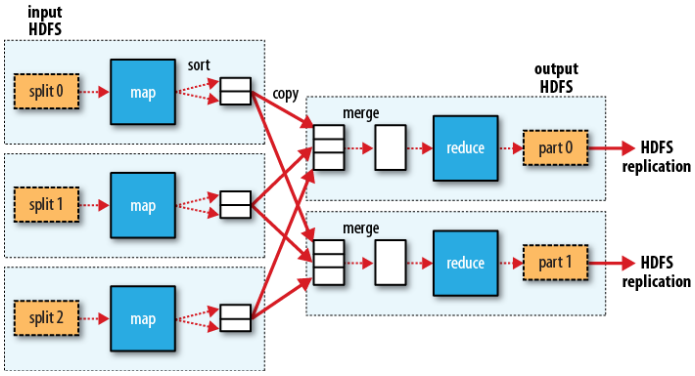

Fig. 1. MapReduce data flow [6]

In this paper we implement MapReduce algorithms and corresponding applications using Hadoop to perform spatial join between trajectory data set and regions around points/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 implementation for processing big trajectory data set over set of POI in the city. Section IV gives the results and presents the evaluation of our implementation. Section V concludes the paper and gives directions for future research.

[1] Dragan Stojanović is with the Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, 18000 Niš, Serbia
E-mail: dragan.stojanovic@elfak.ni.ac.rs
[2] Natalija Stojanović is with the Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, 18000 Niš, Serbia
E-mail: natalija.stojanovic@elfak.ni.ac.rs

II. Related Work

Big data is currently the hottest topic for data researchers and scientists with huge interests from the industry and government agencies [7]. Recently, several initiatives related to Big data have emerged in the European area, such as Euro Lab on Big Data Analytics and Social Mining[1] and the speech "Big data for Europe" held at ICT 2013 Event[2].

The application of high-performance parallel and distributed computing to big spatio-temporal data is of increasing interest at the European level in the European Space Agency[3] (the "Big data from Space" event). At the global level, the Open Geospatial Consortium (OGC) has started activities focused on the geospatial aspects of big data Processing[4], the US government has announced Big data R&D initiative[5], and the industry has already offered commercial solutions, such as the IBM Big data platform[6].

Recently, there is also a growing research interest in spatio-temporal data management, processing analysis and mining using MapReduce model. 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 spatial high level 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.

III. Spatio-temporal Data Processing 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 for a long time period are characterized by 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. 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

[1] www.sobigdata.eu

[2] http://ec.europa.eu/digital-agenda/en/news/big-data-europe

[3] http://www.congrexprojects.com/13C10/

[4] http://www.opengeospatial.org/blog/1866

[5] http://goo.gl/Rp2EZz

[6] http://www-01.ibm.com/software/data/bigdata

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, (t1,t2)>* 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 pattern of her movement.

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*.

IV. 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 have 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 our study, implementation and evaluation 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 1.3 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 20 Map tasks and 10 reduce tasks that finish the job for 1 hour and 9 minutes, producing an output of about 56 MB stored on HDFS. The job output is in the form of records shown in Table I for particular users.

TABLE I
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]
...		
14216	2694	[09.01.2009. 07:00 - 09.01.2009. 09:20]
14216	6264	[09.01.2009. 09:34 - 09.01.2009. 11:04]
14216	788	[09.01.2009. 11:30 - 10.01.2009. 01:00]
…		

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

TABLE II
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
97	1326	72378
132	1317	69428
409	1301	109586
318	1289	121484
276	1247	110236
…		

The results produced can be visually analyzed using a specified tool that supports analytics of spatio-temporal and trajectory data, such as Microsoft SQL Server Business Intelligence Features.

The main objective of our work is not to evaluate the performance of trajectory data processing, since we implement our algorithms on a small, available cluster. Since MapReduce and its Hadoop implementation provides excellent scalability in terms of bigger data sets, as well as larger computing resources, our applications can be easily scaled to more powerful computer nodes than those we used and larger cluster with thousands of machines (e.g. Amazon Elactic MapReduce - EMR) to achieve much higher performances, that are generally expected.

V. 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 in pseudo-distributed mode and on a small cluster of commodity computers, show viability of our approach in usability of MapReduce/Hadoop in big spatio-temporal data processing and analysis.

Although the literature is rich in spatio-temporal data management, processing and mining techniques, handling massive trajectory data with MapReduce is expected to boost the performance of analytic tasks in big trajectory data. 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. Since trajectories change frequently by 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).

VI. Acknowledgments

Research presented in this paper is funded by Ministry of education, science and technological development, Republic of Serbia as part of the project “Environmental Protection and Climate Change Monitoring and Adaptation”, Nr. III-43007.

References

[1] A. Clematis, M. Mineter, and R. Marciano, “High performance computing with geographical data,” Parallel Comput., vol. 29, no. 10, pp. 1275–1279, Oct. 2003.

[2] S. Shekhar, “High performance computing with spatial datasets,” Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems - HPDGIS ’10, pp. 1–2, 2010.

[3] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz, “Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce,” Proceedings VLDB Endowment, vol. 6, no. 11, Aug. 2013.

[4] J. Zhang, “Towards personal high-performance geospatial computing (HPC-G),”Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems - HPDGIS, pp. 3–10, 2010.

[5] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Commun. ACM, vol. 51, no. 1, p. 107, Jan. 2008.

[6] T. White, *Hadoop: The Definitive Guide, 3rd Edition*, O’Reilly Media, p. 688, 2012.

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

[8] A. Cary, Z. Sun, V. Hristidis, and N. Rishe, “Experiences on Processing Spatial Data with Using MapReduce in Practice”, Proceedings of 21st International Conference on Scientific and Statistical Database Management, pp. 302-319, 2009.

[9] K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, and X. Song, “Accelerating Spatial Data Processing with MapReduce”, Proceedings of the 2010 IEEE 16th International Conference on Parallel and Distributed Systems, pp. 229–236, 2010.

[10] S. Zhang, J. Han, Z. Liu, K. Wang, and Z. Xu, “SJMR: Parallelizing spatial join with MapReduce on clusters”, Proceedings of the 2009 IEEE International Conference on Cluster Computing and Workshops, pp. 1–8, 2009.

[11] Q. Ma, B. Yang, W. Qian, and A. Zhou, “Query Processing of Massive Trajectory Data based on MapReduce,” Proceeding of the first international workshop on Cloud data management - CloudDB’09, pp. 9–16, 2009.

[12] B. Yang, Q. Ma, W. Qian, and A. Zhou, “Truster: Trajectory data processing on clusters,” Proceedings of 14th International Conference Database Systems for Advanced Applications DASFAA, pp. 768–771, 2009 .

[13] A. Eldawy and M. F. Mokbel, “A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data,” Proc. VLDB Endow., vol. 6, no. 12, pp. 1230–1233, Aug. 2013.

[14] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia, “On the Impact of User Movement Simulations in the Evaluation of LBS Privacy- Preserving Techniques,” Proceedings of the 1st International Workshop on Privacy in Location-Based Applications, vol. 397, 2008.