2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing

# An Architecture to Process Massive Vehicular Traffic Data

Simon Kwoczek*, Sergio Di Martino†, Thomas Rustemeyer‡ and Wolfgang Nejdl§
*Group Research, Volkswagen AG, Wolfsburg, Germany, Email: simon.kwoczek@volkswagen.de
†DIETI, University of Naples "Federico II", Naples, Italy, Email: sergio.dimartino@unina.it
‡Eckcellent IT, Dortmund, Germany, Email: thomas.rustemeyer@eckcellent-it.de
§L3S Research Center, University of Hanover, Hanover, Germany, Email: nejdl@l3s.de

***Abstract*—Fostered by the big data hype in mobility, many research efforts have been aimed at improving techniques to model vehicular traffic patterns for mobility prediction. Nevertheless, from a practical stance, the industry still faces many technological challenges in bringing solutions on the market. Especially the scalability and performance of such systems raise major concerns, given the amount of spatio-temporal data to be processed. The common approach in dealing with these issues is to introduce constraints and/or simplifications on both the spatial component of the data and on the employed algorithms, leading to results that are somehow limited. To overcome these issues, in this paper we report on our experiences and our approaches in providing a solution that meets industrial needs with the aim to leverage the computational and storage capabilities of the Cloud to handle massive dataset for providing vehicular traffic predictions. In particular, we present an approach to deal with real-world datasets to facilitate the knowledge discovery process from this data while matching the business constraints given by the industrial use case.**

## I. Introduction

Besides being a key source of pollution and stress factor for road users, traffic congestions cost billions of dollars to the society every year. A report from the McKinsey Global Institute estimates that the use of smarter technologies able to help vehicles in avoiding congestions and reducing idling at red lights could lead to savings up to about $600 billion annually by 2020 [1].

As a consequence, many research efforts have been devoted world-wide to improve traffic predictions. Latest developments within the Intelligent Transportation Systems (ITS) community, based on sophisticated sensing and communication technologies (like Smartphones, GPS handhelds, etc.), has led to a significant increase in the availability of mobility datasets with high spatial and temporal resolution. Based on that, many data-driven solutions have been developed that use these datasets to extract new knowledge for modeling future mobility situations, leading to interesting predictive performances [2]–[4]. Nevertheless, most of these solutions based their experiments on so called "small world scenarios" whereby they focus their investigations on either limited geographic regions (e.g.: [5], [6]) or on data for limited time spans (e.g.: [7], [8]). While this approach is acceptable for the given research scenarios, solutions in an industrial context need to be based on architectures and algorithms able to easily handle a broader spatial and temporal coverage.

One representative example is the service of vehicular traffic prediction. This services requires, for each geographic region of interest, (I) to collect data from real-time traffic providers, (II) to create an historical dataset, and (III) to analyze this dataset to learn patterns and predict future traffic situations. The total amount of data to be managed is impressive, since companies providing real-time traffic information have reached a significant spatio-temporal coverage of the road network, world wide. Today, they are able to offer data even in areas where no sensors in the infrastructure are available, since they are mainly collecting Floating Car Data (FCD) [9]. The resulting volume of spatial data (both historical and real time) for a geographic region like a nation, where the situation about hundreds of thousands of road segments has to be constantly monitored, exceeds by far the capability of traditional storing systems. Moreover, since data is collected continuously, preprocessing, aggregating and storing the received data streams on time in a scalable manner leads to great challenges in terms of computational resources required to provide predictive results on time.

These challenges are currently being targeted, in a general manner, within the Big-Data research community, especially from those research activities about so called *Spatial Big Data* (SBD). In [10], authors highlight that " [...] the size, variety, and update rate of [mobility] datasets exceed the capacity of commonly used spatial computing and spatial database technologies to learn, manage, and process the data with reasonable effort". Moreover, they point out that new cloud computing paradigms, tentatively called *spatial cloud computing*, could be a viable solution for these problems. Some generic technological solutions to deal with SBD are presented in [11]–[16], but, to the best of our knowledge, to date there is no specifically tailored solution or architecture being proposed, able to tackle the spatio-temporal complexity of these massive mobility datasets.

In this paper, we report on our experiences and our approaches in providing a solution to store and analyze massive amounts of traffic data from an industrial provider in order to enable the Knowledge Discovery process able to scale up to any geographic level, thanks to an intrinsic parallelizability of the solution. More in details, the contributions in this paper can be summarized as follows:

1) we propose an architecture to store, analyze and visualize massive amounts of traffic data from a commercial provider;
2) we propose two different storage solutions based on relational and NoSQL data models;

978-1-4673-9473-4/15 $31.00 © 2015 IEEE
DOI 10.1109/3PGCIC.2015.124

3) we provide preliminary benchmarks on the reference implementation of the architecture;

The remainder of this paper is structured as follows: In Section II we describe the problem at the hand, also in terms of data sources and potential use cases. In Section III we describe the parallel architecture we propose to tackle the requirements posed by the massive datasets and the use cases. In Section IV we provide a detailed description of two potential implementations of the data models, using a relational database and a NoSQL one. Then, in Section V we present some preliminary results of the performance comparison between the two data models. Finally, in Section VI some concluding remarks and future research directions are pointed out.

## II. The Problem Description

In order to understand the motivations underlying the proposed architecture, as a first step we describe the problem and the typical use cases to be faced in order to provide traffic predictions. Then we will present the type and the size of the data we are dealing with.

The problem we are facing is to provide world-wide traffic predictions, based on mobility patterns that are automatically learned from real traffic data. For this class of problems, the typical use case is to process a dataset of historical traffic information in order to learn some recurring patterns over some road segments. To this aim, we start by describing the employed data sources, then we exemplify two possible use cases.

### A. The Data Sources

To learn traffic trends in order to perform prediction, two main types of data sources are needed:

1) A map of the area to be investigated (potentially world-wide), and
2) A continuous stream of real time traffic information (potentially world-wide), which has to be stored in order to generate a historical dataset.

Actually it is possible to find on the market many providers and many formats for this kind of data. In the following we describe the data types we used in our experiments.

*1) The Digital Map:* The map data contains a detailed description of the road network for the region of interest. In our case, we adopted a commercial automotive solution based on the Navigation Data Standard (NDS)[1], which is a global physical storage standard employed by many players in the automotive domain. Within this map format, each road is divided into several segments, also named *links*, whose length varies from a few meters (mostly at intersections and in urban scenarios) up to a couple of kilometers. The road segments are connected through nodes, called *intersections*, to form a graph. Usually each road segment has information about its geographic position, shape, type of road (highway, urban, etc.), allowed driving directions and further more. For our experiments we employed the map of the entire Germany, which is about 2GB and contains millions of links.

*2) The Traffic Dataset:* As for the real traffic information, we employed two different data sources, with a different level of spatial and temporal granularity:

- The Traffic Flow (TF) dataset covers traffic information for major road segments. The data received uses a table-based referencing system, called Traffic Message Channel (TMC), where there is a fixed number of predefined road segments identified by an ID, for each nation. Each traffic message from TMC, which can be also broadcasted via FM radio, includes the IDs of the involved segments plus some information about the current speed on that stretch (in km/h). Since there is a totally different level of spatial granularity between TMC segments (coarse grained) and NDS links (fine grained), a decoding step is required to match the TMC message onto an actual position on the digital map. As an example, the TMC table for entire Germany contains only 56,000 segments and is updated twice per minute.
- The Traffic Flow Detailed (TFD) dataset comes from a commercial traffic provider and contains the same kind of information as the TF dataset, but with a significantly better spatial and temporal precision. It is based on an dynamic referencing system called OpenLR[2], which covers major and side roads of the street network. Again, as an example, the TFD dataset for Germany contains about 750,000 locations and is updated once a minute. To match these data onto a road network, an open source software to decode OpenLR binary codes can be used[3]. This dataset leads to a continuous massive stream of traffic data, whose storage can pose severe challenges to standard database solutions.

### B. Use Case Description

To define an architecture able to leverage the vast amounts of traffic data, both in terms of spatial and temporal dimensions, we need to have in mind what could be the possible use cases. Two example use cases, which are intended as external software modules interacting with our system, are an *Analyzer* and the *Visualizer* components. We selected these use cases because they are quite at the extremes of a spectrum of functionality required to the underlying architecture, having contrasting requirements.

The *Analyzer* Actor is meant to run data analytic methods for selected road segments within a given spatial region on rather long temporal spans (i.e. extract the traffic model for Wednesdays for one selected road segment using 52 weeks as input data). Indeed, to learn the historical traffic patterns, clearly it is not feasible to execute some Machine Learning algorithms on the whole datasets, but data must be filtered to reach a manageable size. Once the problem is reduced at a reasonable size, then some learning algorithms can be executed [17]. This use case requires to perform spatio-temporal queries on the datasets over limited geographic regions but for long time periods. As a consequence, the underlying storage layer must support a fast execution of this kind of queries.

[1] http://www.nds-association.org/

[2] http://www.openlr.org/

[3] http://www.openlr.org/decoder.html

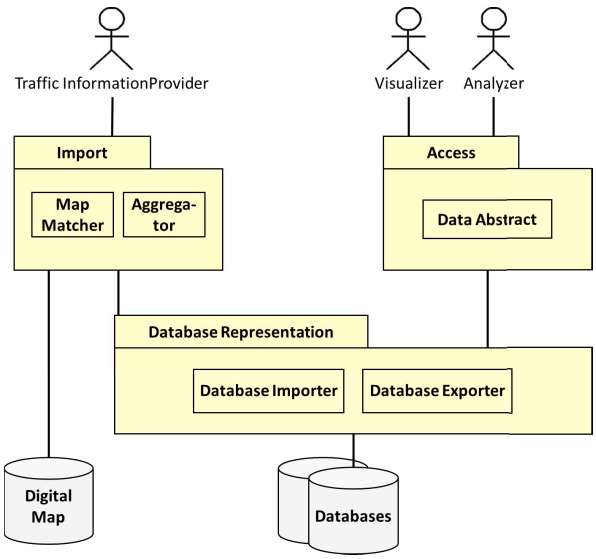


Fig. 1. Component view of the system architecture

The *Visualizer* Actor is intended to show or also replay historic traffic situations in a selected geographic area and time span. It is mainly used by human operators to understand the dynamics of mobility by visualizing traffic information over a short period of time in a potentially rather big geographic area. An example of this use case could be to visualize the traffic in Berlin, Germany during rush-hour on a given day. The use case, especially the replay functionality, generation of charts or heat maps, requires fast data access to traffic data for huge amounts of selected locations over relatively small time spans.

## III. The Proposed Architecture

Analyzing traffic data with the goal to learn patterns and predict future traffic situations is a highly complex task. In this section we provide a description of the architecture we propose by introducing the overall system design, describe the data preparation process and introduce the system interaction. The system is composed of three main components, named *Import*, *Access* and *Database Representation*, as shown in Figure 1. In the following we provide a detailed description for each of these components.

### A. The Import Component

The *Import* component is responsible for receiving, preprocessing and storing the massive amount of incoming data. Data packets (in the following called "messages") are received by an online connection using the Protocol Buffer[4] format. Each of these messages contains information about the traffic situation for a specific stretch of the road (encoded in a map-independent referencing system) at a given timestamp *t*. After decoding the message, the *Import Component* starts the preprocessing pipeline that mainly consists of two sub-steps:

1) *Map Matching* is required due to the map-independent referencing format of the different traffic data types (as introduced in Section II). Each map-independent reference code received is decoded using different open source software components and mapped onto the underlying NDS digital map. With the resulting NDS references a standard geo-line object is calculated that serves as spatial reference in our architecture and is indexed in the database to allow spatial queries.
2) *Spatial Aggregation* describes the process of transforming data into a different aggregation format. As described in Section II, traffic data is updated between one and two times per minute. Each update thereby contains all traffic information for a vast geographic region (mostly nation wide) for the time of the update. However, moving from this temporal aggregation (i.e. traffic information for one minute for entire Germany) to a more spatial oriented (i.e. traffic information for several month for selected street segments) is a key requirement fo many use cases.

Figure 2 illustrates the preprocessing described above. The data is decoded from the received Protocol Buffer format, the location references are map matched, and the resulting data packets are stored in a temporary database. After a predefined time period that data has been processed and stored in the database (in our setup we picked 24 hours as time granularity), the spatial aggregation task begins. Thereby, the entire corpus of data is split up into separate data blocks for each location that hold the traffic information for this specific location for the entire day. These data elements are exported from the temporary database and imported to the final storage solution using the database connector elements within the Database Representation component.

The main focus of the way we designed the preprocessing pipeline is its applicability on cluster or cloud environments. With the data granularity provided, the mentioned preprocessing process can easily be parallelized, also on highly scalable environments, according to some temporal or spatial criteria of the received data.

### B. The Access Component

The *Data Access* act as an access layer between the Database Representation and the external Actors of the system. It receives information from the underlying data structures and provides a unified interface to access traffic data for the use cases. The data is always transformed to a uniform representation used by the Visualizer and Analyzer, so it is totally abstracted from the actual representation in the database.

### C. The Database Representation Component

The *Database Representation* component handles a variety of connectors for different database management systems, which will be introduced in Section IV, that can be used for both the temporary and the final databases. It offers wrapper objects based on different mapping-frameworks for each database setup, including also different storage paradigms, to allow the other layers to easily and transparently handle the underlying data storage.

[4]https://developers.google.com/protocol-buffers

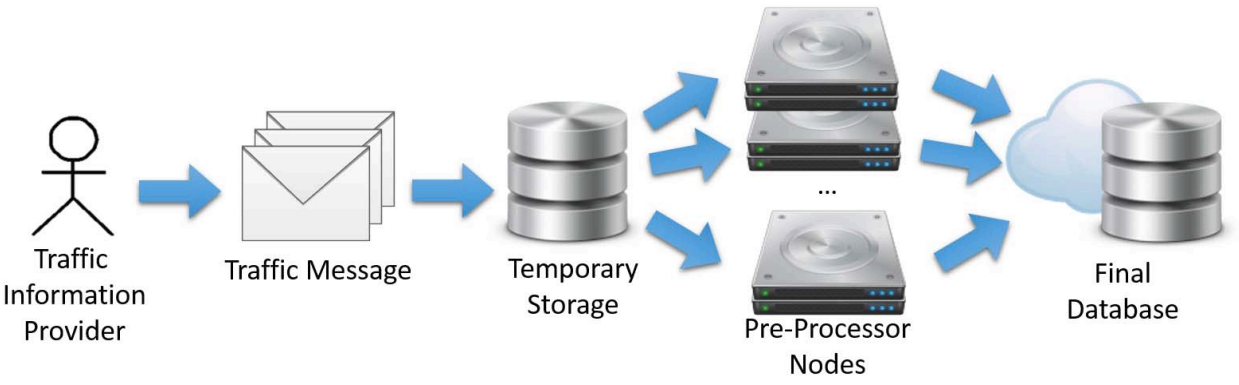


Fig. 2. The pipeline of the parallelized preprocessing

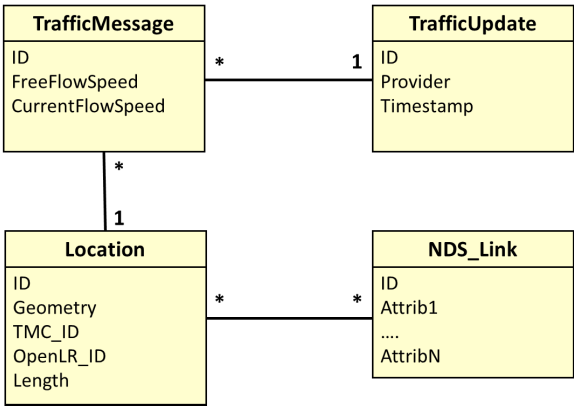


Fig. 3. The designed relational schema for storing traffic information

## IV. The Databases

While there are multiple options for designing an architecture suitable to process traffic data, there are only two main solutions for the data storage: those based on Relational Database Management Systems (RDBMS) and those that base on NoSQL Data Management Systems. In our proposal, we investigated both solutions, trying to understand and measure their pro and cons. In this section we present how we modeled the data in both options.

### A. The Relational Database Model

The first implementation is a relational database and provides a fixed schema for the datasets. The schema we designed to store traffic data is shown in Figure 3, and consists of four different entities: *TrafficUpdate*, *TrafficMessage*, *Location* and *NDS_Link*.

The *TrafficUpdate* is a high level entity that models the concept of an update on the traffic situation received by a traffic provider. It contains an *ID*, the name of the *Provider*, and the *Timestamp*. As described in Section II, each Traffic Update we got at predefined time intervals contains a massive amount (usually nation-wide) of traffic incidents, which are modeled by the following entity. This entity is also used to monitor the import process to recognize duplicate import attempts or fill gaps of missing data at a later time.

The *TrafficMessage* entity is the key table, containing the information about a specific traffic incident. Such an incident is described by an *ID*, the *FreeFlowSpeed* attribute, that represents the drive speed (in km/h) under non-congested conditions and the *CurrentFlowSpeed* attribute, that represents the speed driven (in km/h) at the specific time of the traffic situation. The temporal coverage of the message can be retrieved by the reference to the related tuple in the table *TrafficUpdate*, while its spatial coverage is in the related table *Location*

The *Location* is the spatial reference of the traffic messages. This is a table that could be both pre-filled or that can be built incrementally. Indeed, it contains the geometries for each of the spatial descriptor (TMC for the TF dataset or OpenLR for the TFD one) of a traffic message. Since TMC and OpenLR locations are fixed and known a priori, this table could be precompiled at the first run of the system. Of course in this case there is the risk to have a huge amount of useless information, in case there are some stretches of road where traffic never happens. The other solution is to add a tuple to this table every time we get a message for a Location which is not already in the DB. We opted for this second solution. As for the attributes of this table, we have an *ID*, the *Geometry* (as a LineString) and the *Length* of the stretch in meters.

Finally, the *NDS_Link* table comes from the proprietary NDS format, which allows data analysis of links contained in different Traffic Messages. This entity is essential in order to use generated traffic predictions for applicable navigation solutions, since the predicted traversal time of each link is the key information used by Route Calculation algorithms.

The described relational database model allows easy access of traffic information, both in terms of spatial and temporal queries. For example, using the *geometry* attribute of the *Location* table, we can use spatial queries within our database to filter out locations that traffic information exist for in a geographic area of interest. Similarly, a query on the *timestamp* attribute of the *TrafficUpdate* table allows us to filter the dataset only for a give time frame. In our system, we implemented this schema using the H2 DBMS[5].

[5]http://www.h2database.com

```
Location{
    String ID
    String tmc
    String openlr
    LineString geometry
    List {
        String nds_ref
    }
}
```

```
Traffic{
    Location_ID
    Long timestamp
    Double freeflow
    List {
        Long time
        Double current_speed
    }
}
```

Fig. 4. Example dataset for storing data in NoSQL databases

### B. The NoSQL Data Management Model

The second instance of our architecture is more Cloud-Oriented, being based on a NoSQL database, namely MongoDB. The so called "NoSQL" database systems are a set of (mostly) open source databases, that are explicitly designed to run on distributed environments. In contrast to traditional RDBMS, they use a completely different data model, can operate without putting the data into a fixed schema, such as tables and references, and are thereby highly scalable. On the downside, they mostly lack true ACID transactions but instead provide a concept called "eventual consistency" whereby a lack between changes of data and it's propagation might exists. Our Proposal is build upon two different collections within the NoSQL Database, called *Location* and *Traffic*.

The *Location* collection holds all information about the spatial units that traffic data is available for. It is structured in a similar way as the *Location* entity in IV-A but it also contains the references to the underlying map references.

The *Traffic* collection holds the actual traffic information for an entire day, together with a timestamp and the id of the location that the data is referencing to. The *freeflow* attribute holds the information about the normal travel speed on the referenced location on that particular day. This information comes directly from the data provider. The traffic data is stored in a list of timestamps (minute of day) and the current_speed (in km/h) during that timestamp.

This simplified data model is possible because NoSQL databases typically support higher level data structures like lists (or arrays), maps and sub-documents, even with the possibility to index them. For example the list of NDS references in the *Location* can be indexed and queried efficiently.

## V. Preliminary Experimental Results

In order to understand if there are differences in the scalability of the proposed technological solutions for the data layer, we conducted a set of preliminary experiments. In particular, we analyzed the performance for the two main tasks that all use cases have in common: *Data Insertion Performance* and *Data Extraction Performance*. For that benchmark, all experiments were conducted on the same hardware setup to ensure the comparability of the results.

### A. Data Insertion Performance

As described in Section II, to predict traffic situation there is the need to constantly store, for further processing, the stream of real-time traffic data, which in our case is updated each minute. As a consequence, in order to process this vast amount of mobility data in reasonable time, efficient insertion of the received data packets is a key aspect. We have measured the performance of the two DBMS with respect to the insert functionality, with different amounts of data. As for the Relational Database, we have some B-Tree indexes on part of the attributes used for the queries, that have to be updated for each data entry. As a consequence, as we can see from the Figure 5, for each insert, there is a quite linear relationship between the required time and the number of traffic messages already in the database.

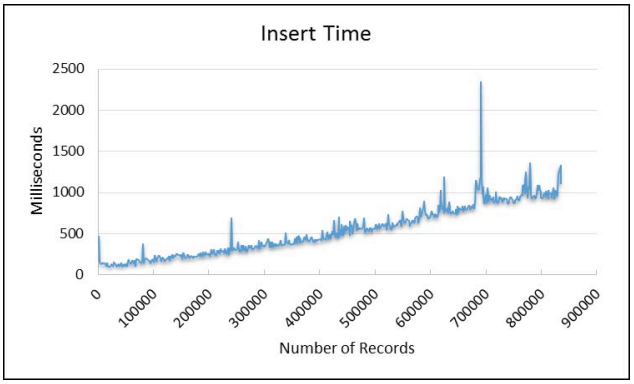


Fig. 5. Experimental time for record insert in the RDBMS

As for the NoSQL database, since the amount of entities is highly decreased due to simplified data model and insertion strategy for whole days, the insertion time stays constant even for very large data sets, which is comparable to 20 to 30 milliseconds per 1000 messages. In our example, we tested it for a year of traffic data based on TMC locations.

### B. Data Extraction Performance

After the data is preprocessed and stored in the underlying storage system, efficient query times are crucial for most use cases. A good example is the described *Visualizer* use case. To replay a historic traffic situation, queries for those traffic situations need to be very fast. Figure 6 shows the results of our benchmark analysis between the query times of the H2 Database and the NoSQL Database. The x-Axis shows the amount of locations that we queried traffic data for an entire day for and the y-axis shows the resulting query time. The results show that the H2 RDBMS is quite slow in providing the first results, probably due to the fact that is Java-based and thus the pipeline is longer than with MongoDB, which is written in C++. Nevertheless, from Figure 6 we can see that for both the solutions there is an interesting stability and scalability, since the performance are only slightly degrading, even in presence of significant changes in the amount of processed data.

## VI. Conclusions and Future Work

One of the greatest challenges mobility providers are facing today is to alleviate the social end economic negative effects of traffic congestions. In times of the big data hype, lots of research is currently being devoted, within the Intelligent Transportation Systems, to develop data-driven solutions. However, to the best of our knowledge, only few research attention has been dedicated to a world-wide scalability of

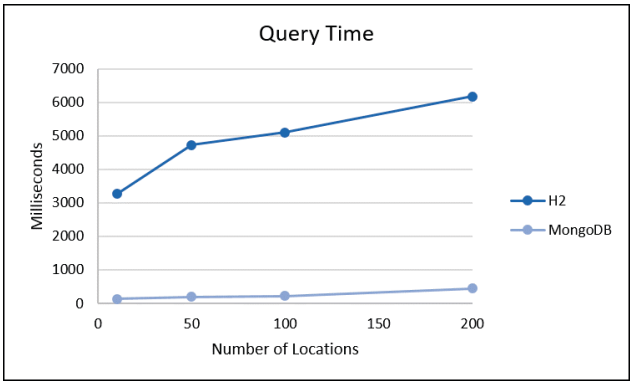


Fig. 6. Experimental time for record extraction from both databases

these proposals, leveraging the findings and the proposals of the Spatial Big Data community.

To investigate this research area, we have proposed a parallelizable architecture characterized by a high degree of scalability. Indeed, all the needed preprocessing steps can be performed in a distributed environment, such as a cluster or the Cloud, and thus being able to process the massive amount of data that characterize the problem. Moreover, we have implemented and tested two different storage solutions, based respectively on a relational and on a NoSQL data model. In a preliminary experimental assessment of performances, we have shown that both the data models are able to tackle the query of the data. However, given the massive amount of incoming traffic data updates, the RDBMS show a lack of performance in managing these continuous updates, not being able to scale as easily as the NoSQL solution.

As future research direction, other than providing a broader investigation of the performance of the architecture under different use cases, it will be interesting also to test new emerging technologies. Especially the newly emerged solutions from the Spatial Big Data community, like for instance Spatial Hadoop [18], should be evaluated for the considered use cases.

## ACKNOWLEDGMENTS

This research has been partly funded by the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement n 610990 - Project COMPANION.

## REFERENCES

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute, Tech. Rep., 2011.

[2] S. Kwoczek, S. Di Martino, and W. Nejdl, "Predicting and visualizing traffic congestion in the presence of planned special events," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 973–980, 2014.

[3] B. Pan, U. Demiryurek, C. Shahabi, and C. Gupta, "Forecasting spatiotemporal impact of traffic incidents on road networks," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, Dec, pp. 587–596.

[4] S. Kwoczek, S. D. Martino, and W. Nejdl, "Stuck around the stadium? an approach to identify road segments affected by planned special events," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*.

[5] S. Kwoczek, S. Di Martino, and W. Nejdl, "Predicting traffic congestion in presence of planned special events," in *Proceedings of the Twentieth International Conference on Distributed Multimedia Systems*, ser. DMS, 2014, pp. 357–364.

[6] B. Pan, U. Demiryurek, and C. Shahabi, "Utilizing real-world transportation data for accurate traffic prediction," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 595–604.

[7] S. Dunne and B. Ghosh, "Regime-based short-term multivariate traffic condition forecasting algorithm," *Journal of Transportation Engineering*, vol. 138, no. 4, pp. 455–466, 2012.

[8] J. Z. Zhu, J. X. Cao, and Y. Zhu, "Traffic volume forecasting based on radial basis function neural network with the consideration of traffic flows at the adjacent intersections," *Transportation Research Part C: Emerging Technologies*, 2014.

[9] H. Kargupta, J. Gama, and W. Fan, "The next generation of transportation systems, greenhouse emissions, and data mining," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1209–1212.

[10] S. Shekhar, V. Gunturi, M. R. Evans, and K. Yang, "Spatial big-data challenges intersecting mobility and cloud computing," in *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*. ACM, 2012, pp. 1–6.

[11] D. C. Cugler, D. Oliver, M. R. Evans, S. Shekhar, and C. B. Medeiros, "Spatial big data: Platforms, analytics, and science," *GeoJournal*, 2013.

[12] S. Di Martino, C. Giorio, and R. Galiero, "A rich cloud application to improve sustainable mobility," in *Web and Wireless Geographical Information Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6574, pp. 109–123.

[13] F. Amato, A. Mazzeo, V. Moscato, and A. Picariello, "Exploiting cloud technologies and context information for recommending touristic paths," in *Intelligent Distributed Computing VII*. Springer, 2014, pp. 281–287.

[14] M. R. Evans, D. Oliver, K. Yang, X. Zhou, and S. Shekhar, "Enabling spatial big data via cybergis: Challenges and opportunities," *a book chapter in CyberGIS: Fostering a New Wave of Geospatial Innovation and Discovery (Ed. S. Wang and M. Goodchild)*, 2014.

[15] 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 of the VLDB Endowment*, vol. 6, no. 11, pp. 1009–1020, 2013.

[16] R. R. Vatsavai, A. Ganguly, V. Chandola, A. Stefanidis, S. Klasky, and S. Shekhar, "Spatiotemporal data mining in the era of big spatial data: algorithms and applications," in *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*. ACM, 2012, pp. 1–10.

[17] S. Shekhar, M. R. Evans, J. M. Kang, and P. Mohan, "Identifying patterns in spatial information: A survey of methods," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 193–214, 2011.

[18] A. Eldawy and M. F. Mokbel, "Spatialhadoop: A mapreduce framework for spatial data," in *Proceedings of the IEEE International Conference on Data Engineering (ICDE'15). IEEE*, 2015.