Real-time streaming mobility analytics

András Garzó*, András A. Benczúr*, Csaba István Sidló*, Daniel Tahara†, Erik Francis Wyatt‡
* *Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI)*
Faculty of Informatics, University of Debrecen and Eötvös University, Budapest
Email: {garzo, benczur, scsi}@ilab.sztaki.hu
† *Yale University*
Email: daniel.tahara@yale.edu
‡ *St. Olaf College*
Email: wyatte@stolaf.edu

***Abstract*—Location prediction over mobility traces may find applications in navigation, traffic optimization, city planning and smart cities. Due to the scale of the mobility in a metropolis, real time processing is one of the major Big Data challenges.**

In this paper we deploy distributed streaming algorithms and infrastructures to process large scale mobility data for fast reaction time prediction. We evaluate our methods on a data set derived from the Orange D4D Challenge data representing sample traces of Ivory Coast mobile phone users. Our results open the possibility for efficient real time mobility predictions of even large metropolitan areas.

***Keywords*-Mobility, Big Data, Data Mining, Visualization, Distributed Algorithms, Streaming Data**

I. Introduction

Intelligent Transportation is at the center of worldwide transport policies intending to consolidate efficient and sustainable transport systems and associated infrastructures. The belief is that smart systems can receive, manage and provide valuable information that will allow transport users and operators to deal with a vast variety of traffic situations: congestion, safety, tolling, navigation support, law enforcement, as well as environmental sustainability.

Real time traffic prediction, as opposed to offline city planning, requires processing the incoming data stream without first storing, cleaning and organizing it in any sense. Scalability and low latency are crucial factors to enable any future technology to deal with mobility traces. This situation pushes towards new algorithms (typically, approximate or distributed) and new computational frameworks (e.g., MapReduce, NoSQL and streaming data). In this paper, we show that location prediction algorithms can be implemented in a distributed streaming environment, and remarkably high throughput can be achieved with low latency using a properly designed streaming architecture.

We use the D4D Challenge Data Set[1] for our experiments. In our research the emphasis is on the algorithmic and software scalability of the prediction method. Although there exist publications with similar goals, even recent results [1] consider data sets of similar or smaller size compared to D4D. Furthermore, we multiplied the original data set to meet the requirements of a metropolitan area of several million people using mobile devices all day.

The rest of this paper is organized as follows. Section II is devoted to describing the streaming data processing software architecture. Section III shows how distributed mobility data stream processing can be implemented in this architecture using Storm. Section IV describes the D4D data set used for our experiments. In Section V we describe the elements of the modeling and prediction framework. In Section VI we give our results, both in terms of accuracy and scalability. Finally related results are summarized in Section VII.

II. Streaming Architecture

Figure 1 depicts the layered architecture of our proposed general mobility data processing solution. The system enables easy model implementation while relying on the scalability, low latency and fault tolerance of the underlying distributed data processing software.

Distributed stream processing frameworks have not yet reached the same level of maturity such as batch processing ones based on MapReduce and key-value stores. Certain mature frameworks such as Hadoop [18] reach the required level of scalability, but cannot provide mechanisms for streaming input and real time response. As it is yet unclear which programming model of distributed stream processing will reach the same maturity and acceptance, we build an abstract stream processing layer capable of using some of the competing alternatives. We indicated Storm and S4 in Figure 1 as the most promising ones.

Stream processing frameworks cannot directly guarantee to store history information as their processing modules may restart from scratch after failures. For example if a machine crashes that stores information on part of the users in memory, these users will be repartitioned to other machines with empty history by default. To ensure that the history is preserved over the processing nodes even in case of failures, we build a generic persistence module (bottom right side of Figure 1). We store information on users and cell towers needed for modeling in distributed key-value stores. The

The publication was supported by grant OTKA NK 105645, the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE No 288956), and the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project is supported by the European Union, co-financed by the European Social Fund. D. Tahara and E Wyatt participated in the project during their study abroad semester at AIT-Budapest http://www.ait-budapest.com/.

[1]http://www.d4d.orange.com/home

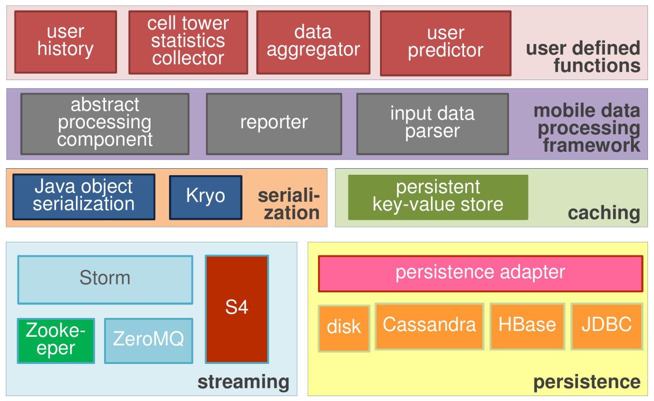

Figure 1. Layers of our mobility prediction architecture: the streaming framework (bottom left), persistence components (bottom right), and the custom analytics (top).

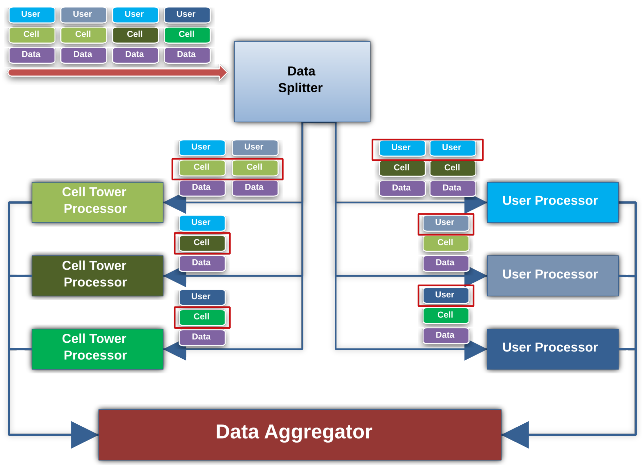

Figure 2. Sample input data partitioning in the streaming framework. Input records consist of tuples of user, cell tower and time stamp and may get partitioned both based on user and cell ID. User and cell based models may get merged through the data aggregator element of the streaming framework.

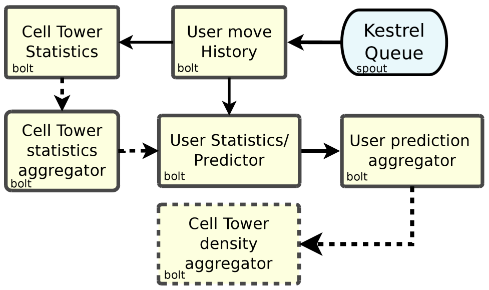
Cell Tower Statistics
bolt
User move History
bolt
Kestrel Queue
spout
Cell Tower statistics aggregator
bolt
User Statistics/ Predictor
bolt
User prediction aggregator
bolt
Cell Tower density aggregator
bolt

Figure 3. System block diagram of the Storm streaming components. Regular arrows show normal while dashed show the low frequency periodic special “tick” packets.

visualization dashboard also gets the required data through the storage adapter.

We defined a modular architecture for caching and serialization. Since near real time processing is very important, we deploy Cassandra [11] due to its high throughput writing capabilities and memcached [7] for its efficiency.

The mobility data processing layer (second from top in Figure 1) provides domain-specific primitives. For example, parsers, abstract data records and processing components for call detail records (CDRs) and other types of input are defined here. Built on these primitives, on the top of Figure 1, user defined functions implement history collection and location prediction. On the top level of our architecture, the implementation details of distributed processing, persistence, caching and serialization are hidden from the programmer to enable agile and straightforward model implementation.

III. Distributed Location Prediction Implementation

Our demonstration is based on Storm, a scalable, robust and fault tolerant, open source real time data stream processing framework developed and used by Twitter and many others [12]. Key to a practical application, Storm takes all the burden of scalability and fault tolerance.

We implement the required processing elements using the predefined abstract components of the Storm API: *spouts* responsible for creating input and *bolts* containing the processing step. Storm can distribute and balance multiple instances of spouts and bolts in the cluster automatically. Bolts can be connected to each other in acyclic processing graph by data packet streams as seen in Fig. 3.

Raw mobility data is read into the memory by an external application and is put into a lightweight message queuing system called Kestrel [4]: Storm spouts get their data from this buffer.

Key to our application is that partitioning can be controlled. As seen in Fig. 2, we may split incoming records both by user and by cell tower. Hence we may define processing components both on the user and on the cell tower base. Finally user and cell tower models can be merged by using data aggregators.

We define two types of data flow, as seen in Fig. 3:

- One regular packet starts from the single spout for each input record that spreads along the thick edges in Fig. 3.
- Periodic aggregation updates move model information along the dashed edges initiated by the special built-in Storm *tick* packets.

We describe our algorithms by specifying the type of data moving along different edges of Fig. 3 and describing the algorithms implemented within each node of the Figure, the bolts that correspond to the steps described in Section V.

- The spout element emits tuples (u, a, t) of user, cell and time stamp.
- User history elements send sequences of (a, t) tuples of the past steps both to the last cell statistics bolt for recording the new user location and to the previous cell for counting frequencies through the cell.
- User history elements send trees rooted at the current

location (a, t) weighted with transition probabilities.

- Cell statistics elements periodically submit the frequent patterns to a single cell statistics aggregator bolt.
- The cell statistics aggregator bolt periodically refreshes the cell frequent patterns to all user statistics predictors.
- User statistics predictors emit the aggregated future history of the user in a form of rooted trees. This element is used in the current experiment to measure the accuracy of the user location prediction.
- User prediction aggregator periodically emits the predicted density of all cells seen in the prediction of the given user for aggregation by the single cell density aggregator element. In the current experiment this element measures the accuracy of the cell density prediction.

IV. The D4D Data Set

We used the Fine Resolution Mobility Trace Data Set (SET2) of the D4D Challenge [2], containing trajectories of randomly sampled individuals over two week periods. Trajectories have the granularity of cell tower locations. Only locations associated with voice, text or data traffic events are provided.

The SET2 data contains 50 million events. In a day, a large metropolis is expected to generate records two to three orders of magnitude more, especially if all locations related to all communication with the base stations is provided. Our target is to process events in the order of 100,000 in a second corresponding to several million people, each generating an event in every minute.

The fine-grained D4D data set is sparse to protect privacy. To reach the targeted amounts of data we merged the two week chunks of user location samples and considered the resulting data as if it is coming from a single two weeks period. The resulting weekly aggregated traffic volume is shown in Fig. 5, indicating that considering the time of the day only may be a good approximation for user motion.

The fact that only two-week user histories are available in the data set poses certain limitations for our experiments, however provides realistic distributions for scalability testing. In the data set the median users only visits three locations, and the mean only visits six. The median user generates 46 events, out of which changes location thirteen times. Most calls are in the same location as the previous call as seen in Fig. 4. We can achieve near 70% accuracy by always predicting the user's last location. In addition, we should only ever predict locations that a user has visited before and since we cannot see a smooth path for how a user moves over time, for now we ignore the physical layout of the antennas and treat location prediction as a classification problem.

V. Models for Location Prediction

We give sample models to predict user movement and traffic congestion. We produce a simple yet realistic framework for location prediction sufficiently complex for scalability testing. We predict sequences and evaluate always for the next known location after the predefined prediction period.

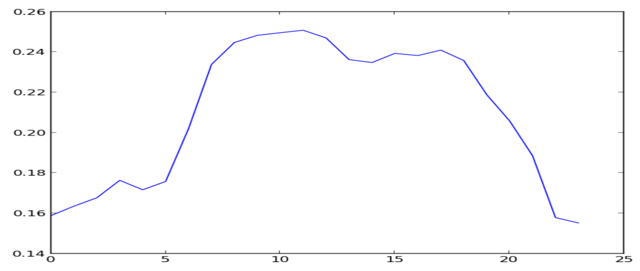

Figure 4. Fraction of calls that are at a different antenna than the previous call for that user (y axis) by time of day (x axis). We can see morning and evening rush hour, and people move less at night. Even at the peak of morning rush hour, more then three quarters of all calls are from the same location as before.

The main component of our model is based on learning the patterns of individual user motion histories. Our main assumption is that for most users, their location follows a daily regular schedule, i.e. they leave to work and return home at approximately the same time of the day over roughly the same route. This assumption is confirmed for other data sets e.g. in [9]. We consider typical locations and two-step frequent patterns. For each user, we generate the following set of features:

- Time of the day;
- Time elapsed since the previous location;
- Ratio of events from frequently used antennas;
- Typical locations at the time of the day and distance from previous location;
- Typical length of stay in one place in general and depending on time of the day.

The last two classes of features are implemented by nearest neighbor search among blocks of events consisting of subsequent events from the same location. Distance is calculated by taking the time of the day, the duration of stay at the same location, the geographical distance and the equality of the present and the past antennas. We compute the nearest two neighbors under four different selection of these attributes:

(A) Time of the day only;
(B) Time of the day and equality of the past cell tower;
(C) Duration of stay and distance from the previous cell tower;
(D) Time of the day, duration of stay and distance from the previous cell tower.

Based on the above feature set, we use decision trees for modeling. First we predict whether the user changed or remained in the same location. For location prediction we have no information other than user past locations and frequent paths through user most recent locations. Hence we train classifiers separate for each of the following possibilities for the next location:

- Same as previous location;
- Most frequent (second most frequent) location;

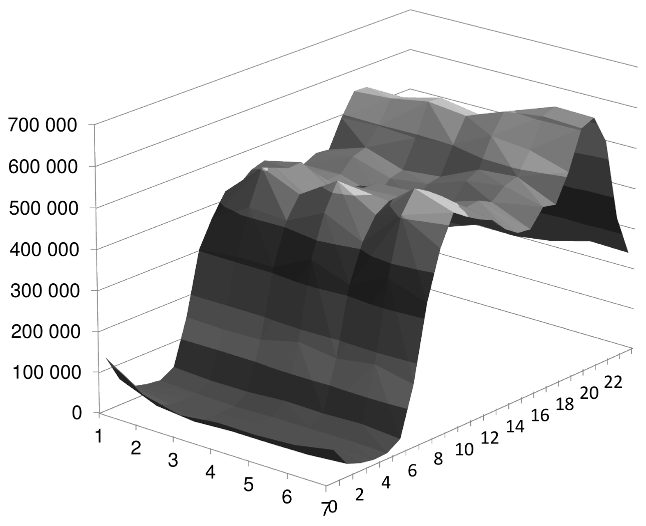

Figure 5. Volume of traffic (vertical axis) as the function of the day of the week (1–7) and hour (0–23) over the horizontal axes.

- One of the nearest neighbor locations;
- Next step over frequent paths—here longer paths could also be computed, e.g. by streaming FP-trees [8].

We consider the first week as training period: for each user event, we compute all the above features based on the user history before the event. We give a set of binary labels to the event to mark whether the user stayed in the previous location or moved to one of the potential new locations. As additional features, we also compute the physical distance of the last location to each of the potential new ones.

In our implementation, the modeling steps correspond to the Storm bolts of Fig. 3 as follows. User features are computed based on past history in the `user move history` bolt. In order to compute frequent paths, the `cell tower statistics` bolt receives the last few user steps from the `user move history` bolt. Frequent paths need only be periodically updated and this is done in the `cell tower statistics` bolt that feeds the `user statistics/predictor` bolt with updates. This bolt is capable of implementing the pre-trained decision tree model.

VI. Experiments

In this section we describe our measurements for speed, scalability and quality. To emphasize scalability in the number of threads and machines, we ran our experiments over a Storm 0.9.0-wip4 cluster of 35 old dual core Pentium-D 3.0GHz machines with 4GB RAM each.

The spouts emit as many records as the Storm framework is able to process. We partitioned the data for the spouts and for each spout, we loaded its data set into memory while initializing the topology. We iterated over the data infinitely, i.e. the same user moves were emitted repeatedly.

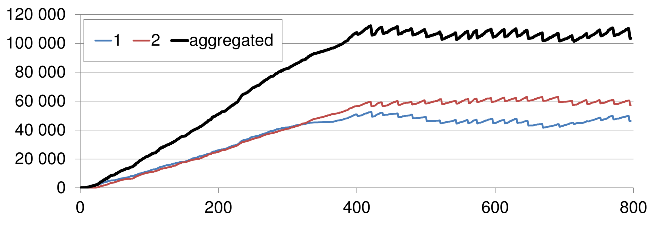

Figure 6. Number of records emitted by two spouts per 13 minutes (vertical axis, records per second) after initializing the topology (with seconds on the horizontal axis). Red and blue lines indicate throughput of two spouts and the black bold line is the aggregated speed.

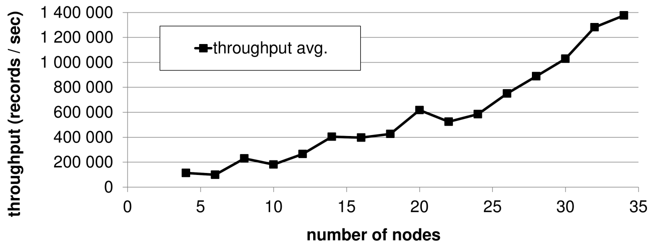

Figure 7. Throughput (number of records per second) as the function of the number of servers in the Storm cluster, with five input spouts residing at five different servers.

A. Scalability and Latency

To test scalability of location prediction we test how the throughput (the number of events processed per second) changes when new nodes are added to the cluster. To avoid misleading figures due to caching, we ran the system for 10 minutes before starting to measure the predictor element processing rate. Figure 6 shows how system throughput normalizes after initialization.

Figure 7 depicts throughput speed. Near linear scalability can be observed in the number of servers and threads: We may reach rates of a few 100,000 records in a second, which is well beyond the desired threshold.

The average latency of the system was low, processing an input record took about 1023 ms. We did observe larger values when initializing the system, but this value remained relatively constant when adding or removing nodes.

B. Fault Tolerance

When a node fails, a new node is initialized with the stored states of the affected processing components. According to the guarantees of Storm, the lost packets are also processed again. Figure 8 shows how node failures affect overall performance. We can observe rapid recoveries, despite of the large number of failing nodes, the overall performance remains predictable.

C. Accuracy

Next we evaluate the accuracy of our location prediction methods by giving F-measure (averaged for the positive and negative classes) and AUC values. We predict the location of active users for at least 15 minutes in the future. We

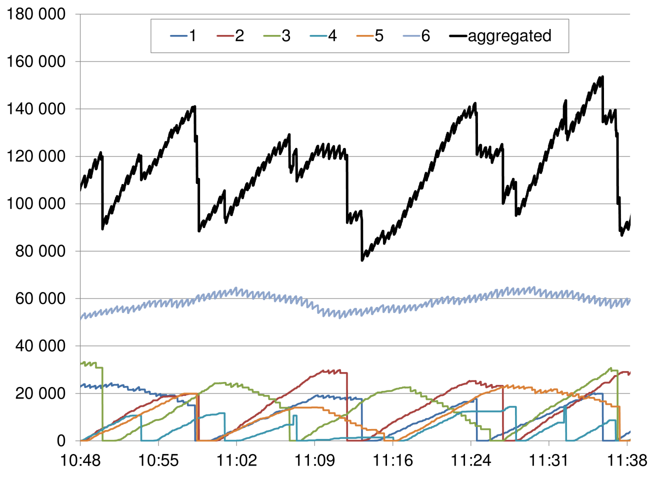

Figure 8. Throughput (number of records per second) as the function of the time passed (absolute times), with six nodes, each with a spout. One node works continuously, while the others occasionally stop.

	All		Active	
	F	AUC	F	AUC
next	0.888	0.914	0.888	0.886
15 mins	0.859	0.905	0.819	0.896
1 hour	0.861	0.909	0.815	0.890

Table I
ACCURACY OF THE PREDICTION WHETHER A GIVEN USER MAKES THE NEXT CALL FROM A NEW LOCATION.

consider a user active if he or she has at least 1000 events during the two-week period. There are 1126 such users in the data set. We use the first week of data for all users for training and evaluate over the second week.

The prediction for users staying in place is given in Table I. Here we observe that the prediction quality is very high and slightly decays as we look farther ahead in the future. The decision tree has 37 nodes using the following sample of attributes in approximate order of tree depth:

- Previous location equal to most frequent user cell;
- Fraction of the last and the most frequent cells in the user history so far;
- Geographical distance and duration of stay at nearest neighbor (D) and other nearest neighbors in case of the active users;
- Elapsed time since arrival to the last location.

The prediction for the next user location is evaluated for active users for a minimum of 15 minutes in the future. As seen in Table II, we perform binary classification tasks for ten different types of likely next locations for the user. Note that some of these locations may coincide or not exist, hence no multi-class classification is possible. Based on the measured accuracy of the methods and the likelihood assigned by the decision tree, it is easy to merge the binary predictions into a prediction of a single location.

The decision for the most frequent continuation of the last two cell locations is weak, however misclassification

	F	AUC
same as previous	0.862	0.896
most frequent 3-step trajectory	0.372	0.623
nearest neigbor (A)	0.637	0.686
second nearest neigbor (A)	0.633	0.633
nearest neigbor (B)	0.710	0.699
second nearest neigbor (B)	0.700	0.695
nearest neigbor (C)	0.708	0.705
second nearest neigbor (C)	0.698	0.695
nearest neigbor (D)	0.553	0.686
second nearest neigbor (D)	0.672	0.669

Table II
ACCURACY OF THE PREDICTION FOR DIFFERENT TYPES OF NEW LOCATIONS AS DESCRIBED IN SECTION V.

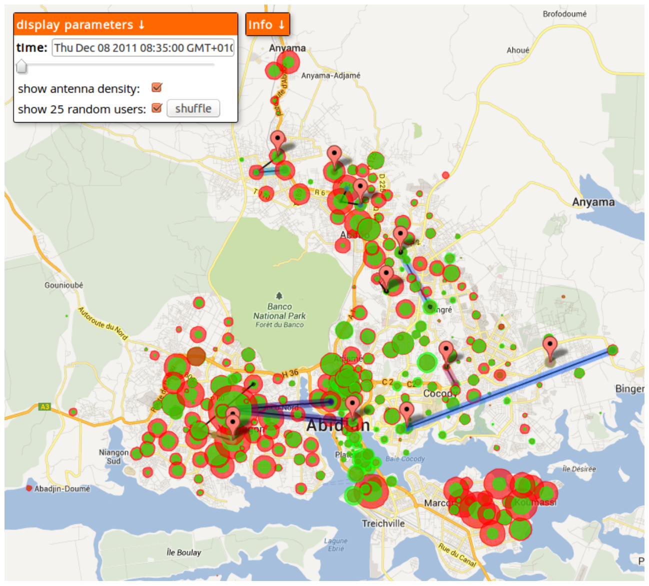
display parameters ↓
info ↓
time: Thu Dec 08 2011 08:35:00 GMT+010
show antenna density: ✓
show 25 random users: ✓ shuffle

Figure 9. Visualization of the cell traffic prediction (red circles show actual sizes while green is the prediction), with a sample of individual movement predictions (black lines are real, colored lines are predicted moves).

is imbalanced: we almost never misclassify users who do not follow the frequent path. Here the decision tree is surprisingly small, it has only five leaves. The first decision splits whether the user stayed for more or less than one day at the same location. Subsequent decisions use the fraction of the previous cell among all cells of the user and the distance of the last step taken.

For nearest neighbors, the decision trees mostly choose based on physical distance. This is the main reason why we see very similar measures for the last eight classification problems. In addition, some features to determine whether the user stays in place are also used but in general the decision trees are much smaller than for staying in place.

We developed a visualization demo application to demonstrate the use of individual trajectory predictions: Fig. 9 shows the aggregated predicted and real cell density as well as the predicted and real trajectories of random users.

VII. Related Results

The idea of using mobility traces for traffic analysis is not new. Early papers [14] list several potential applications, including traffic services, navigation aids and urban system mapping. In [14] a case study of Milan, while in [1] of New York City suburbs are presented.

Mobility, City Planning and Smart City are considered by many major companies. IBM released redguides [10], [15] describing among others their IBM Traffic Prediction Tool [15]. Unfortunately little is known about the technology and the scalability properties of existing proprietary software.

Several recent results [9], [16], [3], [17], [19] analyze and model the mobility patterns of people. In our results we rely on their findings, e.g. the predictability of user traces.

We are aware of no prior results that address algorithmic and software issues of the streaming data source. Mobility data naturally requires stream processing frameworks [12], [13], [5]. A wide range of stream processing solutions are available: For example, major social media companies have all developed their software tools [6].

VIII. Conclusions and Future Work

In this preliminary experiment we demonstrated the applicability of data streaming frameworks for processing mass mobility streams: Low latency and high throughput values enable building real-time applications based on motion prediction. Given more detailed data, our framework is suitable for detecting flock (motion of groups), deviation of real track from expected (map) or permitted (restricted areas) tracks. Our results open the ground for advanced experimentation regarding the quality of large scale mobility prediction suitable for example for real time traffic prediction.

Acknowledgment

Among several colleagues who draw our attention to the scalability issues of mobility data and inspired our research, we are especially grateful, in alphabetic order, to Tamas Bodis (Vodafone Hungary), Csaba Fekete (NSN Hungary), Zalan Heszberger (Ericsson Hungary and TU Budapest), Attila Medvig (NNG) and Janos Tapolcai (TU Budapest).

References

[1] R. A. Becker, R. Caceres, K. Hanson, J. M. Loh, S. Urbanek, A. Varshavsky, and C. Volinsky. A tale of one city: Using cellular network data for urban planning. *Pervasive Computing, IEEE*, 10(4):18–26, 2011.

[2] V. D. Blondel, M. Esch, C. Chan, F. Clérot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. Data for development: the D4D challenge on mobile phone data. *CoRR*, abs/1210.0137, 2012.

[3] de Montjoye, Yves-Alexandre, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Nature Sci. Rep.*, 2013.

[4] E. D. Dolan, R. Fourer, J.-P. Goux, T. S. Munson, and J. Sarich. Kestrel: An interface from optimization modeling systems to the neos server. *INFORMS Journal on Computing*, 20(4):525–538, 2008.

[5] M. Dusi, N. d'Heureuse, F. Huici, A. di Pietro, N. Bonelli, G. Bianchi, B. Trammell, and S. Niccolini. Blockmon: Flexible and high-performance big data stream analytics platform and its use cases. *NEC Technical Journal*, 7(2):103, 2012.

[6] D. Eyers, T. Freudenreich, A. Margara, S. Frischbier, P. Pietzuch, and P. Eugster. Living in the present: on-the-fly information processing in scalable web architectures. In *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, page 6. ACM, 2012.

[7] B. Fitzpatrick. Distributed caching with memcached. *Linux journal*, 2004(124):5, 2004.

[8] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining*, 212:191–212, 2003.

[9] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[10] M. Kehoe, M. Cosgrove, S. Gennaro, C. Harrison, W. Harthoorn, J. Hogan, J. Meegan, P. Nesbitt, and C. Peters. Smarter cities series: A foundation for understanding ibm smarter cities. *Redguides for Business Leaders, IBM*, 2011.

[11] A. Lakshman and P. Malik. Cassandra: A structured storage system on a p2p network. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 47–47. ACM, 2009.

[12] J. Leibiusky, G. Eisbruch, and D. Simonassi. *Getting Started With Storm*. Oreilly & Associates Incorporated, 2012.

[13] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 170–177. IEEE, 2010.

[14] C. Ratti, S. Williams, D. Frenchman, and R. Pulselli. Mobile landscapes: using location data from cell phones for urban analysis. *ENVIRONMENT AND PLANNING B PLANNING AND DESIGN*, 33(5):727, 2006.

[15] S. Schaefer, C. Harrison, N. Lamba, and V. Srikanth. Smarter cities series: Understanding the ibm approach to traffic management. *Redguides for Business Leaders, IBM*, 2011.

[16] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[17] L. A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C.-C. Hung, and W.-C. Peng. On discovery of traveling companions from streaming trajectories. In *ICDE*, pages 186–197, 2012.

[18] T. White. *Hadoop: The Definitive Guide*. Yahoo Press, 2010.

[19] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang. On discovery of gathering patterns from trajectories. In *ICDE*, pages 242–253, 2013.