# Large-scale Online Mobility Monitoring with Exponential Histograms

Christine Kopp
Fraunhofer IAIS
St. Augustin, Germany
christine.kopp@iais.fraunhofer.de

Michael Mock
Fraunhofer IAIS
St. Augustin, Germany
michael.mock@iais.fraunhofer.de

Odysseas Papapetrou
Technical University of Crete
Chania, Greece
papapetrou@softnet.tuc.gr

Michael May
Fraunhofer IAIS
St. Augustin, Germany
michael.may@iais.fraunhofer.de

## ABSTRACT

The spread of digital signage and its instantaneous adaptability of content challenges out-of-home advertising to conduct performance evaluations in an online fashion. This implies a tremendous increase in the granularity of evaluations as well as a complete new way of data collection, storage and analysis. In this paper we propose a distributed system for the large-scale online monitoring of poster performance indicators based on the evaluation of mobility data collected by smartphones. In order to enable scalability in the order of millions of users and locations, we use a local data processing paradigm and apply exponential histograms for an efficient storage of visit statistics over sliding windows. In addition to an immediate event centralization we also explore a hierarchical architecture based on a merging technique for exponential histograms. We provide an evaluation on the basis of a real-world data set containing more than 300 million GPS points corresponding to the movement activity of nearly 3,000 persons. The experiments show the accuracy and efficiency of our system.

## 1. INTRODUCTION

Advertising media are under the obligation to provide reliable performance indicators for the pricing of advertising campaigns. For the German out-of-home (OOH) advertising industry, generating yearly net sales of about 760 million Euro [2], this has meant to establish a system of geographically differentiating performance indicators over the past years[1]. However, with the spread of digital signage also a fine-grained *temporal* differentiation will be required in future. While current performance indicators inform about the poster contacts of seven or ten average days (the two standard durations of poster campaigns in Germany), digital out-of-home (DOOH) advertising spots have a duration of only a few seconds. Assuming an evaluation period of 10 seconds, the granularity of the performance indicators (and consequently of the required input data) increases by four orders of magnitude. DOOH therefore has to face the challenge of collecting and analyzing *big data*. In addition, digital content has the advantage that it can be instantly adapted to a changing audience. This adaptation, however, requires *online* performance information, which forms the second challenge of DOOH performance evaluation.

In this paper we propose a distributed system for the large-scale online monitoring of poster performance indicators based on the evaluation of mobility data collected by smartphones. We hereby consider two use cases which the system shall cover. First, we want to be able to perform online queries which obtain performance measures for the recent past in a sliding window style. Second, we want to analyze historic data for various time intervals. The first type of query allows the online monitoring of poster performance and thus the targeted placement of advertisement spots. The second type of query can be used for billing purposes or to analyze previously collected data sets (e.g. to find interesting visit patterns that can then be monitored in the online system). Although our use cases differ with respect to their system requirements (distributed online processing vs. analysis of massive amounts of centralized data), we want to keep the maintenance effort of the system as low as possible. Our goal is therefore to set up a scalable system architecture that allows an efficient re-use of code from the online scenario for historic data analysis.

The key component of our approach to handle massive streams of data is to use exponential histograms for data compression. This data structure has the advantage that it offers sliding window query capabilities with a guaranteed maximum relative error. In addition, exponential histograms can be applied in a distributed setting [12] thus allowing for scalability when the number of users increases.

Our online system relies on an Android implementation

[1] http://www.agma-mmc.de/media-analyse/plakat.html

that we have used in previous work [3] to detect visit patterns on mobile phones. For the analysis of historic data we have set up a Storm environment. In combination with the Kafka messaging system we are able to perform historic data analysis in a distributed streaming fashion. In this way we can apply the same system architecture for online and historic data analysis. We use the Storm/Kafka environment to perform the experiments in this paper.

We analyze the performance of our system using a real-world GPS data set containing trajectories of 2,967 persons containing more than 300 million GPS points over a period of one week. We extract visit events from this data set using 400,988 points of interest (POI) in Germany from OpenStreetMap (OSM). Our experiments show that the usage of exponential histograms results in an average error of less than 1/10 of the maximum acceptable error while reducing the storage space to an amount as small as 9.7% of the baseline storage space.

The remainder of our paper is organized as follows. Section 2 discusses related work. Section 3 shows our system architecture and Section 4 provides the experiments. We conclude our paper in Section 5.

# 2. RELATED WORK

## 2.1 Exponential Histograms

Exponential histograms [1] are a deterministic structure, proposed to address the basic counting problem, i.e., for counting the number of true bits in the last $N$ stream arrivals. They belong to a family of methods that break the sliding window range into smaller windows, called buckets or basic windows, to enable efficient maintenance of the statistics. Each bucket contains the aggregate statistics, i.e., the number of arrivals and bucket bounds, for the corresponding sub-range. Buckets that no longer overlap with the sliding window are expired and discarded from the structure. To compute an aggregate over the whole (or a part of the) sliding window, the statistics from all buckets overlapping with the query range are aggregated. For example, for basic counting, aggregation is a summation of the number of true bits in the buckets. A possible estimation error can be introduced due to the oldest bucket inside the query range, which usually has only a partial overlap with the query. Therefore, the maximum possible estimation error is bounded by the size of the last bucket.

To reduce the space requirements, exponential histograms maintain buckets of exponentially increasing sizes. Bucket boundaries are chosen such that the ratio of the size of each bucket $b$ with the sum of the sizes of all buckets more recent than $b$ is upper bounded. In particular, the following invariant is maintained for all buckets $j$: $C_j/(2(1+\sum_{i=1}^{j-1} C_i)) \leq \varepsilon$ where $\varepsilon$ denotes the maximum acceptable relative error and $C_j$ denotes the size of bucket $j$ (number of true bits arrived in the bucket range), with bucket 1 being the most recent bucket. Queries are answered by summing the sizes of all buckets that fully overlap the query range, and half of the size of the oldest bucket, if it partially overlaps the query. The estimation error is solely contained in the oldest bucket, and is therefore bounded by this invariant, resulting in a maximum relative error of $\varepsilon$.

Recently, Papapetrou et al. [12] showed how an arbitrary number of exponential histograms $EH_1, EH_2, ..., EH_n$ (each one corresponding to an individual stream) can be aggregated/merged, in order to produce a single exponential histogram $EH_\oplus$ that corresponds to the order-preserving union of the streams. More precisely, let $\varepsilon$ denote the maximum error parameter of the original exponential histograms, and $\varepsilon'$ the parameter of the merging algorithm. The algorithm supports the creation of an aggregated exponential histogram with a maximum relative error of $(\varepsilon + \varepsilon' + \varepsilon \cdot \varepsilon')$. In this work we use this merging algorithm to reduce the memory required for storing the exponential histograms of the visit events coming from various input sources.

## 2.2 Distributed Evaluation of Visit Events

In previous work we have provided a set of visit quantities that can be used to define performance measures in OOH advertisement [8]. In this paper we concentrate on the evaluation of *gross visits* which state the number of total visits to a certain location and which can be used to estimate the total contacts to a poster site. In addition, we have provided a methodology for the privacy-preserving, distributed collection of visit quantities in previous work [7].

The basic idea of the approach is to decentralize the data collection and evaluation process of movement data. Instead of constantly submitting location information of a user to some central server, the evaluation of visits (or visit patterns) is performed *locally* on a mobile device (e.g. smartphone). The device submits only aggregated and anonymized statistics to a central coordinator. In addition, web anonymization techniques such as onion routing [4] can be used to prevent that the coordinator reconstructs visit histories from several messages of a person based on the communication protocol. A similar, however analytically less powerful framework has previously been proposed by Hoh et al. [6] for the distributed, privacy-preserving monitoring of traffic. However, both papers do not consider the practical aspect of scaling the proposed method to thousands and potentially millions of users. In fact, considering movement statistics from our GPS data set, every person traverses more than 200 street segments per day. If we assume further that each person visits 10 different locations (e.g. work location, shops, bus stops) per day and 20 million persons participate in data collection, about 4.2 billion events occur every day. In order to cope with this number of events, sophisticated analysis and storage algorithms as well as a sophisticated system architecture have to be devised. The design and performance analysis of such a system is the scope of our paper.

# 3. SYSTEM ARCHITECTURE

Our architecture consists of two or alternatively three layers (see Figure 1). The lowest layer holds the user nodes, which collect the users' GPS data and extract visit events. The visit events are forwarded either directly to the central coordinator (flat setting) or to a layer of intermediate nodes (hierarchical setting). In the flat setting, the coordinator aggregates the visit information of each POI in an exponential histogram. I.e., for each POI an exponential histogram is maintained that records the visit events for this POI. As the exponential histogram stores a time aggregate with the event, queries over time windows can be answered. In the hierarchical setting, the exponential histograms reside already at the intermediate nodes. In regular time intervals the intermediate nodes submit the exponential histograms to the coordinator, which merges them and answers user queries.

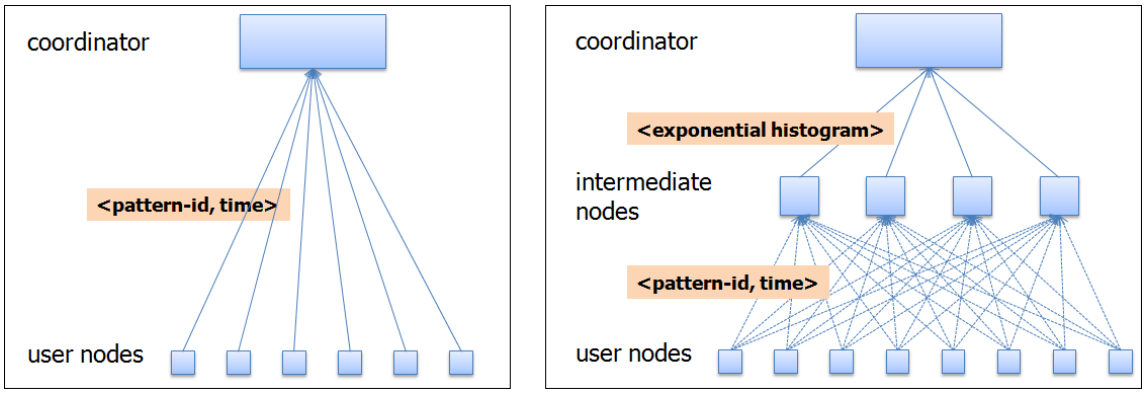



**Figure 1: System architecture; left: flat setting; right: hierarchical setting**

The exponential histograms cannot be applied at the user level because the number of visit events per user is too small to make the data structure efficient. The layer of intermediate nodes was introduced for horizontal scalability and to avoid an overload of the coordinator. However, it also serves a privacy purpose given that the intermediate nodes do not collude (see [7]). As a user can freely select an intermediate node when submitting a visit event, no intermediate node will obtain the whole event history of a single user. The intermediate nodes submit their data structure in regular time intervals to the coordinator, which finally merges the data structures and answers user queries.

As motivated by our use case, our system shall be able to perform analyses online as well as on historic data. The above architecture describes the online use case. For historic data analysis we have to substitute the layer of local nodes. This substitution should still allow to process data in parallel in order to scale to large amounts of data. In addition, a streaming environment would be preferable in order to re-use existing code. Both aspects can be met by using a distributed streaming processing system as, for example, Storm[2] or S4 [11]. We have ported the Android code of event detection to run as Storm bolts. The input is streamed into the system via the Kafka messaging system [9], which allows to handle each GPS point of the recorded trajectories as individual message. Thus, with this mechanisms we can scale the parallel simulation of event detection horizontally in the cluster. In our experiments described in the next section we used this technique to emulate the event detection on GPS traces of 2,967 test persons in an experimental cluster. Detected events are sent to the intermediate nodes similar to the online setting.

# 4. EXPERIMENTS

## 4.1 Data Set

For our experiments we use a subset of a large-scale GPS survey [10] commissioned by the Arbeitsgemeinschaft Media-Analyse e.V.[3], a joint industry committee of German advertising vendors and customers. The GPS data has been collected in the year 2011 and contains 2,967 persons with valid GPS data. The persons are recruited from 31 major cities in Germany and are asked to carry the GPS devices for one week.

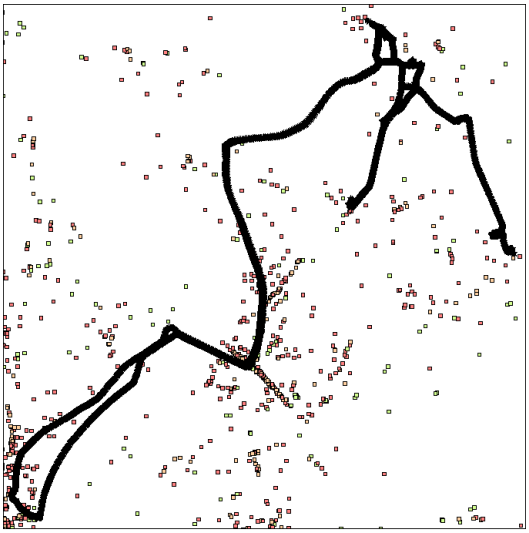
After clean-up the data set contains 304 million GPS points. In addition, we extracted 400,988 points of interest (POI) from OpenStreetMap[4] (OSM) [5] marked with the keys *shop, amenity, leisure, tourism, historic, sport, public transport, railway*. We grouped the POI into the following categories: shop, restaurant, leisure, education, parking and public transport stops. We limited our experiments to those POI because digital posters are still very expensive and therefore placed mostly at attractive places as train stations or shopping locations. For each POI category we defined a minimum stay time and a 50x50 meter spatial buffer in order to extract visit events. Table 1 shows the number of POI aggregated to the six types along with the assumed minimum stay times. Figure 2 left shows a one-day trajectory of one test person along with the extracted POI in its surrounding.

| POI type | # POI | min. stay time |
|---|---|---|
| shop | 89,789 | 10 min. |
| restaurant | 105,665 | 15 min. |
| leisure | 69,318 | 15 min. |
| education | 24,151 | 15 min. |
| parking | 63,602 | 5 min. |
| public transport stop | 48,463 | 5 min. |
| **total** | **400,988** | – |

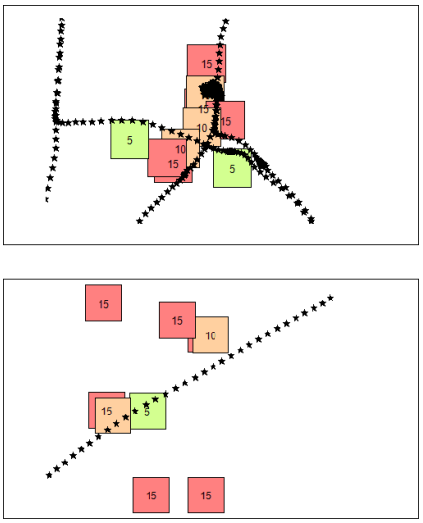
**Table 1: Number of POI extracted from OSM and minimum defined stay time per category**

The extraction of visit events is performed by the *local nodes* (see Section 3). A visit results from the spatial intersection of a trajectory and a geographic location and has to last a given minimum period of time. For a formal definition of a visit see [8]. Figure 2 right shows exemplary the extraction of visit events. The POI are colored according to their minimum required stay time (green = 5 minutes, orange = 10 minutes, red = 15 minutes). In the top right picture one visit occurs in the orange colored POI (where a dense cluster of GPS points exists). In the bottom right picture the user passes the POI merely on his way. As the duration of spatial intersection lies below the minimum stay time, no visit events are generated. For the extraction of visits we apply an algorithm from previous work [3], which

[2]http://storm-project.net
[3]http://www.agma-mmc.de
[4]http://www.openstreetmap.org



(a) one-day trajectory of a test person




(b) trajectory excerpts showing one POI visit on top (dense cluster of points) and two POI passages on bottom

**Figure 2: left: one-day trajectory of a test person along with OSM points of interest colored according to minimum stay time (green/orange/red = 5/10/15 minutes); top right: visit in POI with 10 minute stay time; bottom right: passages of POI without visiting**

| # visits per POI | # POI |
|---|---|
| 1 | 7,590 |
| 2 | 2,176 |
| 3 | 824 |
| 4 | 458 |
| 5 | 223 |
| 6 | 136 |
| 7 | 101 |
| 8 | 53 |
| 9 | 56 |
| $\geq 10$ | 192 |

**Table 2: Frequency of visits per POI**

was designed to extract visit patterns from a stream of GPS positions online on mobile phones.

In total we extracted 23,508 visit events to 11,809 different POI for all test persons. This number has been considerably below our expectations. Most likely it results from two reasons. First, the number of OSM POI are incomplete. From the online source *http://www.haltestellen-suche.de* we know to expect at least 217,000 stations of public transport in Germany, and also the number of shops in Germany is considerably above the extracted number of POI. Second, GPS signals are typically blocked inside of buildings. As we applied a light-weight event extraction algorithm (that can run on a mobile device), we may have lost a number of visit events.

Table 2 shows an overview of the number of visit events per POI. Most often, only a single visit occurred. This number is quite reasonable given our low number of visits and the independent movement behavior of the test persons.

In order to perform experiments also on a large-scale data set resembling more closely the real-world situation, we replicated the original visit data by a factor of 1,000. We set the time of each such visit by adding Gaussian noise to the current time with $\mu = 0$ and $\sigma = 10{,}000$ seconds.

## 4.2 Experimental Set-Up

In our experiments we conducted point queries in a sliding window fashion. I.e., we queried the number of events per POI in the past $\Delta t$ seconds. The selected query windows were of length 30, 600, 1800, 3600 or 86400 seconds. We performed those queries every 10 minutes (in the hierarchical setting this coincides with the time interval of the force action). For our observation period of one week this resulted in $n_t = 1{,}008$ queries per query window for each of the $n_p = 11{,}809$ visited POI. In accordance with our maximum query interval, we set the sliding window parameter of the exponential histogram to 86,400 seconds in all experiments. Further, we varied the maximum acceptable relative error $\varepsilon$ to take the values 0.01, 0.02, 0.04, 0.08 and 0.16. In the hierarchical setting we used 10 intermediate nodes which submitted their data structures every 600 seconds to the coordinator.

We measured the error for each experiment using the mean absolute percentage error (MAPE), which is defined as follows:

$$MAPE = \frac{\sum_{i=1}^{n_p} \sum_{j}^{n_t} \left| \frac{x_{ij} - \widehat{x}_{ij}}{x_{ij}} \right|}{n_p \cdot n_t}$$

where $x_{ij}$ denotes the true number of visit events at POI $i$ in query window $j$ and $\widehat{x}_{ij}$ denotes the number of events returned from the exponential histogram. In the case of $x_{ij} = 0$ we added a relative error of zero if our estimate was correct ($\widehat{x}_{ij} = 0$) and a relative error of $\infty$ if $\widehat{x}_{ij} \neq 0$. This latter case, however, did not occur. We performed all experiments for the flat and hierarchical setting as well as for the original and multiplied data set.

## 4.3 Results

Figure 3 shows the results for the flat and hierarchical setting of the multiplied data set. The respective numbers are provided in Tables 3 and 4. Note that we display only the results for the multiplied data set because due to the few visit events in the original data set the error was nearly always zero there.

In general, the MAPE is very low, lying with one exception below 1%. For both the flat and hierarchical setting two trends can be observed. First, the MAPE decreases with smaller $\varepsilon$. Second, the MAPE decreases with decreasing size of the query window. The first effect is nearly linear for all query windows and can be expected from the characteristics of exponential histograms. The second is also expected because the error guarantees are given on the size of the sliding window, which was fixed to 86,400 seconds. Accordingly, the error for smaller time intervals has to be lower. However, the effect is linear to the logarithm of the query window sizes, i.e. when increasing the query window, the MAPE increases sublinearly.

When comparing the error between the flat and hierarchical setting, the merge operations result in only a small increase in error.

| query wind. | $\varepsilon$=0.01 | $\varepsilon$=0.02 | $\varepsilon$=0.04 | $\varepsilon$=0.08 | $\varepsilon$=0.16 |
|---|---|---|---|---|---|
| 30 s | 7E-6% | 2E-5% | 5E-5% | 2E-4% | 3E-3% |
| 600 s | 2E-4% | 3E-3% | 0.03% | 0.18% | 0.44% |
| 1800 s | 3E-3% | 0.04% | 0.12% | 0.28% | 0.54% |
| 3600 s | 0.02% | 0.06% | 0.15% | 0.32% | 0.59% |
| 86400 s | 0.06% | 0.14% | 0.28% | 0.44% | 0.79% |
| mem. | 14.5 MB | 8.8 MB | 5.5 MB | 3.4 MB | 2.5 MB |

**Table 3: Mean absolute percentage error and memory usage for flat setting**

| query wind. | $\varepsilon$=0.01 | $\varepsilon$=0.02 | $\varepsilon$=0.04 | $\varepsilon$=0.08 | $\varepsilon$=0.16 |
|---|---|---|---|---|---|
| 30 s | 8E-6% | 2E-5% | 6E-5% | 2E-4% | 3E-3% |
| 600 s | 2E-3% | 3E-3% | 0.03% | 0.18% | 0.45% |
| 1800 s | 3E-3% | 0.04% | 0.12% | 0.28% | 0.64% |
| 3600 s | 0.02% | 0.06% | 0.15% | 0.36% | 0.75% |
| 86400 s | 0.07% | 0.16% | 0.32% | 0.53% | 1.03% |
| mem. | 14.5 MB | 8.8 MB | 5.5 MB | 3.4 MB | 2.5 MB |

**Table 4: Mean absolute percentage error and memory usage for hierarchical setting**

In order to set the MAPE in perspective to the number of visit events, Table 5 shows the average and maximum number of visits per POI and query interval. The average is hereby calculated once for all POI and time slots and once only for those containing at least one event.

The memory usage of the exponential histogram at the end of the observation period is depicted in the last line in Tables 3 and 4. Assuming fixed 32-bit counters, it depends only on $\varepsilon$ and the maximum possible count $N$ in the sliding window of each POI, requiring $O(\frac{1}{\varepsilon}\log N)$ space [1]. As we maintain an exponential histogram for each POI, the required memory depends also linearly on the number of (distinct) visited POI which is, however, constant in our experiments.

| query wind. | avg. events | avg. events $> 0$ | max. events |
|---|---|---|---|
| 30 s | 0.1 | 1.6 | 1,916 |
| 600 s | 2.0 | 12.1 | 2,029 |
| 1,800 s | 5.9 | 32.0 | 2,260 |
| 3,600 s | 11.8 | 60.0 | 3,118 |
| 86,400 s | 270.3 | 709.8 | 36,037 |

**Table 5: Number of average and maximum events per POI and query window in ground truth**

## 4.4 Discussion

Our experiments show that the resultant error is very low. For all settings of $\varepsilon$ the mean error (MAPE) is less than 1/10 of the maximum acceptable error. This is a very good result. Especially we can be sure for small total number of visits that the query results are always correct. For example, setting $\varepsilon = 0.01$ will result in no errors if less than 100 events occur per POI. This is an important characteristic because the visit frequency of POI is right-tailed, containing only few POI with very high frequencies.

Further the experiments show that our setting scales horizontally. By introducing a layer of 10 intermediate nodes, the MAPE was on average 7.5% higher and at most 23% higher than in the flat setting. Both numbers are considerably below the maximum acceptable error as well as the maximum relative error guaranteed for the join of exponential histograms.

Finally, to evaluate the memory usage, we can compare the numbers to the following baseline scenario. Whenever a visit event occurs, the POI identifier and timestamp are stored at the coordinator using two 4 Byte integers. As our sliding window covers only one day, we will assume that we have to store 1/7 of the total visit events. For the original 23,509 events this results in 0.026 MB. For the multiplied data set it results in 25.6 MB. The storage amount for the original events using exponential histograms varied between 0.54-4.7 MB. In this case we did not save on memory. However, using the more realistic multiplied data set with exponential histogram sizes between 2.5-14.5 MB, our experiments require only 9.7-56.6% of the baseline storage space depending on the selected $\varepsilon$.

When extrapolating to the envisioned setting of monitoring 20 million persons generating each 210 events per day on about 6,500,000 distinct POIs in Germany (including the 6,000,000 distinct street segments), just storing the raw event data would result in 31.3 GB memory consumption. This is considerably above the 1.3-7.8 GB required by the exponential histograms (by just taking into account that our memory consumption increases linearly with the number of distinct POIs).

Considering our entire approach including exponential histograms and local evaluation, the storage reduction is even much higher compared to a naive centralized setting where the users submit a GPS position every second to some central coordinator.

Also note that inserting into and querying an exponential histogram almost takes constant time far below a microsecond, which is much faster than searching an event database of raw events.

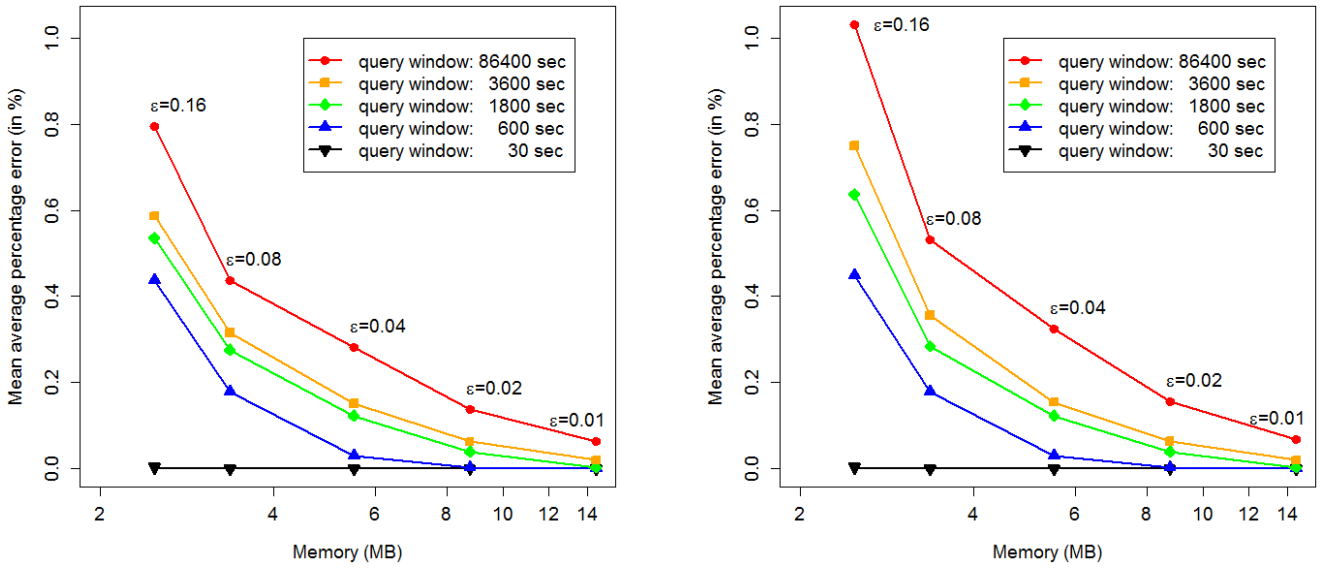



**Figure 3: Mean average percentage error and memory usage for different maximum relative errors ($\varepsilon$) and query window sizes; left: without intermediate nodes; right: hierarchy with 10 intermediate nodes**

## 5. CONCLUSIONS

In this paper we propose a distributed system for the large-scale online monitoring of poster performance indicators based on the evaluation of mobility data. Our system relies on the collection and local processing of mobility data via smartphones and uses exponential histograms for the efficient storage and querying of visit statistics in a sliding window fashion. Our experiments on a multiplied real-world data set with nearly 3,000 persons show that the usage of exponential histograms results in an average error of less than 1/10 of the maximum acceptable error while reducing the storage space to an amount as small as 9.7% of the baseline storage space.

## 6. ACKNOWLEDGMENTS

We thank our colleague Sebastian Bothe for supporting us to run the cluster-based version of the experiments and the Arbeitsgemeinschaft Media-Analyse e.V. for granting the use of the GPS data set. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 255951 (LIFT).

## 7. REFERENCES

[1] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

[2] Fachverband Außenwerbung e.V. Netto-Werbeeinnahmen erfassbarer Werbeträger in Deutschland, 2002-2010 (Net turnover of confirmable advertising media in Gemany, 2000-2010), 2011. `http://www.faw-ev.de/media/download/marktdaten/4\_Nettoumsaetze\_aller_Werbemedien\_ab\_2002.pdf`.

[3] S. Florescu, C. Körner, M. Mock, and M. May. Efficient mobility pattern stream matching on mobile devices. In *Proc. of the Ubiquitous Data Mining Workshop (UDM 2012)*, pages 23–27, 2012.

[4] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Comm. of the ACM*, 42:39–41, 1999.

[5] M. M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[6] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the 6th Int. Conf. on Mobile Systems, Applications, and Services (MobiSys'08)*, pages 15–28. ACM, 2008.

[7] C. Kopp, M. Mock, and M. May. Privacy-preserving distributed monitoring of visit quantities. In *SIGSPATIAL 2012 Int. Conf. on Advances in Geographic Information Systems(SIGSPATIAL/GIS)*, pages 438–441, 2012.

[8] C. Körner. *Modeling Visit Potential of Geographic Locations Based on Mobility Data*. PhD thesis, University of Bonn, 2012.

[9] J. Kreps, N. Narkhede, and J. Rao. Kafka: A distributed messaging system for log processing. In *Proceedings of 6th International Workshop on Networking Meets Databases (NetDB), Greece*, 2011.

[10] Media-Micro-Census GmbH. ma 2012 Plakat - Methoden-Steckbrief zur Berichterstattung, 2012. `http://www.agma-mmc.de/publikationen/methodische-berichte/methoden-steckbriefe.html?eID=dam\_frontend\_push\&docID=179Z`.

[11] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Proceedings of the 2010 IEEE Int. Conf. on Data Mining Workshops*, ICDMW '10, pages 170–177, Washington, DC, USA, 2010. IEEE Computer Society.

[12] O. Papapetrou, M. N. Garofalakis, and A. Deligiannakis. Sketch-based querying of distributed sliding-window data streams. *PVLDB*, 5(10):992–1003, 2012.