FAST: Differentially Private Real-Time Aggregate Monitor with Filtering and Adaptive Sampling

Liyue Fan
Math&CS Department
Emory University, Atlanta, GA
liyue.fan@emory.edu

Li Xiong
Math&CS Department
Emory University, Atlanta, GA
lxiong@mathcs.emory.edu

Vaidy Sunderam
Math&CS Department
Emory University, Atlanta, GA
vss@mathcs.emory.edu

ABSTRACT

Sharing aggregate statistics of private data can be of great value when data mining can be performed in real-time to understand important phenomena such as influenza outbreaks or traffic congestion. However, to this date there have been no tools for releasing real-time aggregated data with differential privacy, a strong and provable privacy guarantee. We propose FAST, a real-time system that allows differentially private aggregate sharing and time-series analytics. FAST employs a set of novel, adaptive strategies to improve the utility of shared/released data while guaranteeing the user-specified level of differential privacy. We will demonstrate the challenges and our solutions in the context of prepared data sets as well as live participation data dynamically collected among the SIGMOD'13 attendees.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data Mining*; G.3 [**PROBABILITY AND STATISTICS**]: Time series analysis

Keywords

Differential Privacy, Estimation, Sampling, Time Series

1. INTRODUCTION

Sharing real-time aggregate statistics of private data enables many data mining applications, such as participatory sensing [8] and data surveillance [6]. Consider use-cases below:

- **Disease Surveillance**: A health care provider gathers data from individual visitors. The collected data, e.g. daily number of Influenza cases, is then shared with third parties, e.g. researchers, in order to monitor and detect seasonal epidemic outbreaks.
- **Traffic Monitoring**: A GPS service provider gathers data from individual users about their locations, speeds, mobility, etc. The aggregated data, e.g. the number of users at each region during each time unit, can be mined for commercial interest, such as popular places, as well as public interest, such as congestion patterns on the roads.

SIGMOD'13, June 22–27, 2013, New York, New York, USA.

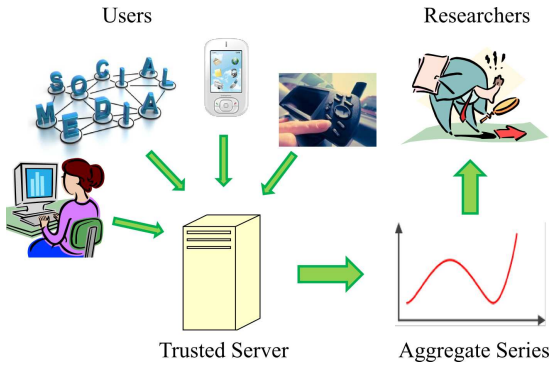


Figure 1: Aggregate Data Sharing Scenario

- **Conference Attendance Tracking**: An hourly count of conference attendees can be aggregated and mined for interesting attendance patterns. Monitoring the shared data, conference participants can also be advised in real-time on the accommodation availabilities and possible traffic congestion around the conference venue.

A common scenario of such applications can be summarized by Figure 1, where a trusted server gathers data from a large number of individual subscribers. The source of data may vary from web searches, mobile devices such as phone and GPS, to social networks. The collected data is then aggregated and continuously shared with other un-trusted parties and researchers for various purposes. The trusted server, i.e. publisher, by contractual obligations must ensure that releasing the data does not compromise the privacy of any individual who contributed data. The goal of our work is to enable the publisher to share useful aggregate statistics continuously (aggregate time series) while guaranteeing privacy.

The current state-of-the-art paradigm for privacy-preserving data publishing is differential privacy [2], which requires that the aggregate statistics reported by a data publisher be perturbed by a randomized algorithm $\mathcal{A}$, so that the output of $\mathcal{A}$ remains roughly the same even if any single tuple in the input data is arbitrarily modified. Given the output of $\mathcal{A}$, an adversary will not be able to infer much about any single tuple in the input, and thus privacy is protected.

There are many challenges in sharing aggregate time series under differential privacy. The first one comes from high correlation of data values at successive timestamps. Applying the standard differential privacy mechanism which adds a Laplace noise [3] to the raw aggregate at each time stamp can lead to a very high overall perturbation error. The utility of shared/released data is greatly impacted in this case. The second challenge is in separating noise from the signal. In other words, given a perturbed value, can we derive a

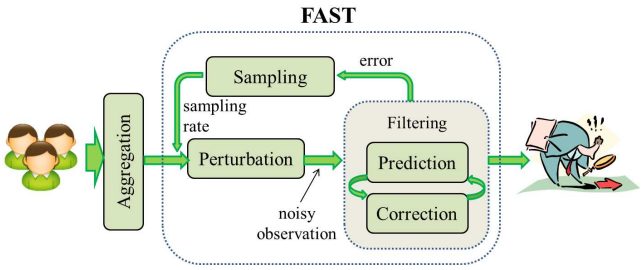


Figure 2: FAST Framework

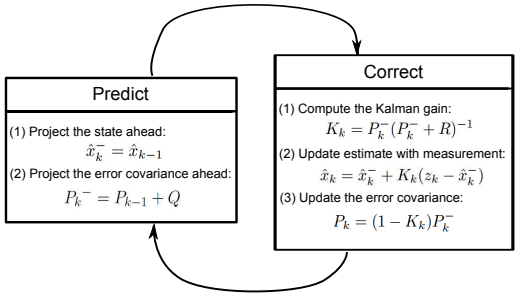


Figure 3: Illustration of the Kalman Filter [5]

good estimate close to the true value? Due to the Laplacian nature of the perturbation noise, there is no analytic form of the posterior distribution. We will have to examine sub-optimal solutions and approximations to derive posterior estimation. The third challenge is to compute and release the estimates in real-time, which is required by all the applications mentioned previously. Methods that require entire data series, such as the Discrete Fourier Transform based algorithm [10], are not applicable to real-time applications.

In this demo, we present FAST, a real-time system with Filtering and Adaptive Sampling for differentially private Time-series monitoring. We implement and extend our recent work [5] and present several contributions. Firstly, FAST provides an adaptive framework to release real-time aggregate statistics under differential privacy. It samples long time-series according to detected data dynamics and uses filtering to predict data values at non-sampling points and to estimate true values from noisy observations at sampling points. The key innovation is that FAST utilizes feedback loops to dynamically adjust the filtering model as well as the sampling rate. Secondly, FAST incorporates a set of novel algorithms to choose from, such as a full range of filtering options and sampling methods. Thirdly, we will demonstrate the system in the context of prepared data sets along with live data dynamically collected from a SIGMOD'13 conference participation survey.

2. SYSTEM OVERVIEW

In this section, we present FAST framework and its key components as well as algorithms. As seen in Figure 2 , the input data is a stream of raw aggregates, collected from individual users, with one value at a discrete timestamp. If sampled by FAST, the aggregate will be perturbed with a calibrated noise by the *perturbation* component to strictly enforce the user-specified level of differential privacy. The *filtering* component utilizes an internal data model to provide prediction at non-sampling points and correction of the noisy/perturbed observation at sampling points. Then the *filtering* estimation error will be fed through the *sampling* component to adaptively adjust the sampling rate. FAST also provides an easy-to-use interface which guides the users through their monitoring tasks. Below we describe each component with technical details.

2.1 Perturbation

The *perturbation* component in FAST acts as a differentially private interface similar to PINQ [9] for any sampled value from the aggregate series, such that the released series satisfies differential privacy. Let α denote the user-specified overall level of privacy, also called privacy budget, and M denote the total number of samples allowed by FAST for a given series or application. A lower value of α implies stronger privacy guarantee and a higher noise. The *perturbation* component distributes α/M budget to each sampled aggregate (i.e. *count*); according to the Composition Theorems [9], the entire released series satisfies α-differential privacy. More formally, given a raw aggregate x_k at time stamp k and the privacy budget α/M, an α/M-differentially private value z_k can be obtained by adding a Laplace noise ν to the raw aggregate [3]:

$$z_k = x_k + \nu, \qquad p(\nu) \sim Lap(0, M/\alpha). \tag{1}$$

We refer to Equation (1) as the measurement model and ν as the measurement noise. The perturbed value z_k is then received as a noisy observation in the filtering module.

2.2 Filtering

The *filtering* module in FAST generates estimates of monitored aggregates in order to improve the quality of released data per time stamp. In our context, "filtering" refers to determining the posterior distribution of x_k, given all measurements z_k's up to k, thus providing an optimal posterior estimate. Two operations, *prediction* and *correction*, will be recursively applied during the monitoring period. The *prediction* step generates a prediction of the data value at each time stamp based on previous release and an internal process model, which provides the temporal correlation between adjacent aggregate values. In our current system, the time series data (i.e. counts) is described by a linear model as follows:

$$x_{k+1} = x_k + \omega, \qquad p(\omega) \sim \mathbb{N}(0, Q) \tag{2}$$

where ω is a white Gaussian noise, also called the process noise. We refer to Equation (2) as the process model which is used by both filters provided by FAST. The *correction* step combines the noisy observation, when available, with the prediction to generate a posterior estimate. The correction mechanism varies according to the filtering method chosen.

FAST provides two filtering options: the Kalman filter [7] and particle filter [1]. The Kalman filter is applicable to linear process model in Equation (2) and it requires Gaussian measurement noise. Therefore, we approximate ν in Equation (1) by a white Gaussian noise with variance R. Thus, the measurement model for the Kalman filter is adapted to:

$$z_k = x_k + \nu, \qquad p(\nu) \sim \mathbb{N}(0, R). \tag{3}$$

In FAST, once users provide the length of input data (T) and the overall privacy budget (α), the variance of the approximate Gaussian noise, i.e. R value, can be computed automatically given the result of our recent study [4]. The advantage of the Kalman filter is that we can easily obtain a minimum variance posterior estimate and the computation cost for each time step is $O(1)$. Figure 3 gives a high-level diagram of the two recursive procedures of the Kalman filter. Note that $\hat{x}_k^-$ denotes prediction, i.e. the prior estimate, while $\hat{x}_k$ denotes correction, i.e. the posterior estimate.

The other filtering option, particle filter, is a widely received approach to non-Gaussian tracking problem. When particle filter is chosen, Equation (1) is adopted for the measurement model and ν

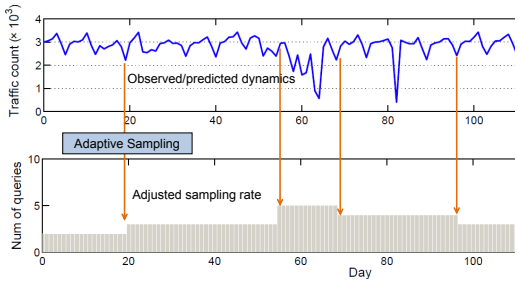


Figure 4: Illustration of Adaptive Sampling

retains the Laplace distribution. In nature, particle filter is a sequential Monte Carlo algorithm and it uses weighted samples (particles) to estimate the distribution of $\hat{x}_k$ at every time stamp, which may not have an analytic representation. Therefore, with a sufficiently large number of particles, denoted as N, the particle filter approaches Bayesian optimal estimate at the cost of computation time, $O(N)$. In FAST, the default value of N is recommended to the user, i.e. $N = 1000$, to achieve a trade-off between accuracy and computational efficiency. We refer to our previous work[4][5] for complete studies about the two filtering methods.

FAST also includes the default option without filtering which directly releases the Laplace perturbed values, i.e. z_k's, at all time points. We denote the default option as LPA in the demonstration.

2.3 Sampling

In order to reduce the overall perturbation error introduced by differential privacy mechanism, FAST samples the input series rather than perturb every data value. The sampling methods implemented in FAST include fixed-rate sampling and adaptive sampling.

The fixed-rate sampling method periodically samples the data series per T_0 time stamps. T_0 denotes the sampling interval and can be determined by the user. If not specified, T_0 is set to 10 in FAST by default. The challenge of fixed-rate sampling is to find the optimal sampling interval *a priori*, which is impractical for most real-time applications.

The adaptive sampling methods in FAST utilize the idea of feedback control and adjust the sampling rate according to detected data dynamics on-the-fly. The feedback is defined as the error between *prediction* and *correction* from the filtering component, measuring how well the internal process model describes the current data dynamics. A small error results in a decrease in the sampling rate and vice versa. FAST adopts PID (*Proportional*, *Integral*, and *Derivative*) control, the most common form of feedback control, and implements P, PI, and PID controllers for users' choice. The *PID* control gains , if not specified by the user, will take on default values as in our previous work [5]. Figure 4 illustrates the idea of adaptive sampling. We refer readers to our work [5] for definitions and the detailed PID algorithm.

3. SYSTEM DEMONSTRATION

FAST is implemented in JAVA with JSC[1] for simulating Laplace distribution. Our demo[2] includes two types of use cases from Section 1: (1) Live participation data collected from SIGMOD'13 attendees in real-time mode; (2) Prepared, complete data sets in batch mode. The utility metric is the relative error between differentially private released values and original aggregates.

[1] `http://www.jsc.nildram.co.uk`
[2] available at `http://www.mathcs.emory.edu/aims/FAST`

3.1 Data sets

SIGMOD'13 Participation Data (Live). In this demonstration, we will engage the SIGMOD'13 attendees to take a conference participation survey and release differentially private hourly count of positive responses. We will post our survey question "Are you at the conference right now?" through secure third party polling tool Poll Everywhere[3] and participants are invited to respond via email, text message, or even through social network sites. The hourly aggregates gathered by the polling tool can then be used as input data in this live demonstration.

In case of technical difficulties or insufficient survey participation, we will also provide a simulated data set that resembles the count of conference attendees over time in order to perform the real-time demonstration.

Prepared Data Sets. We plan to demonstrate FAST with 3 real-world data sets as used in our recent work [5]:

- `Flu` is the weekly surveillance data of Influenza-like illness provided by the Influenza Division of the Centers for Disease Control and Prevention[4]. We collected the weekly outpatient count of the age group [5-24] from 2006 to 2010. This time-series consists of 209 data points.
- `Traffic` is a daily traffic count data set for Seattle-area highway traffic monitoring and control provided by the Intelligent Transportation Systems Research Program at University of Washington[5]. We chose the traffic count at location I-5 143.62 southbound from April 2003 till October 2004. This time-series consists of 540 data points.
- `Unemployment` is the monthly unemployment level of African American women of age group [16-19] from ST. Louis Federal Reserve Bank[6]. This data set contains observations from January 1972 to October 2011 with 478 data points.

As an alternative, we also generated 3 synthetic data sets for demonstration using `linear`, `logistic`, and `sinusoidal` models, all of which contain 1000 data points. The details about the three models can be found in our recent study [4].

3.2 Basic Functionalities

FAST is an easy-to-use system which guides users through their monitoring tasks. FAST interface allows users to enter different parameters, to choose methods according to their needs, and to examine intermediate as well as final results.

We will start the demonstration by introducing the main interface of FAST. From there, users have two options for system mode: real-time release and batch release. The real-time mode enables the user to obtain a private released value immediately after a new, raw aggregate is available, while the batch mode takes a (partial) data series as input and produces a released series. After selecting the system mode, we will guide the audience through the settings of each FAST module. The audience will have an opportunity to specify values for some parameters such as the overall privacy guarantee, i.e. α, and the variance for the process noise, i.e. Q in Equation (2), while the rest parameters can be automatically computed or set to default values. In real-time mode, multiple instances with different parameter settings and options can be run at the same time. In batch mode, multiple sampling methods and filtering techniques are provided for comparison and the utility results will be summarized in the end. For instance, the audience could select and

[3] `http://www.polleverywhere.com/`
[4] `http://www.cdc.gov/flu/`
[5] `http://www.its.washington.edu/`
[6] `http://research.stlouisfed.org/`

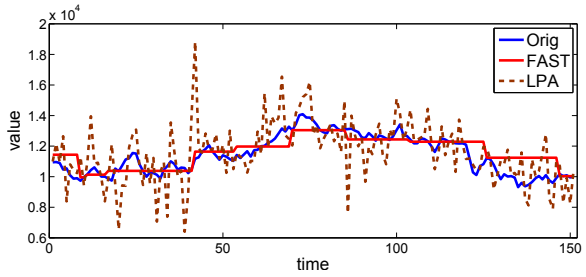
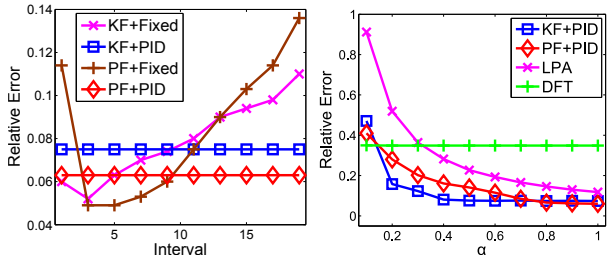


Figure 5: Released Series vs. Original Series



(a) Sampling Comparison (b) Filtering Comparison

Figure 6: Comparisons with Linear Data Set

compare the performance of the Kalman filter, particle filter, and the default LPA algorithm for the filtering component.

3.3 Real-Time

After the settings have been made, users will be able to experience the real-time sharing feature of FAST. We will show that every time when a new, raw, aggregate value is available, a differentially private estimate can be generated in real time. At the end of the monitoring application, summaries of utility and computation time, privacy cost incurred and the number of samples, original as well as released data series etc., will be provided to the users for reference. Figure 5 provides a visualization example of the original `linear` data, the released series by FAST, and that of the default LPA algorithm. With the same privacy guarantee, FAST released series retains much higher accuracy (i.e. data value, trend) than the LPA released series.

Furthermore, we can also demonstrate FAST in debug mode so that interested audience can look under-the-hood to see how filtering and sampling are done per time stamp. The audience could further explore the original data series, intermediate results such as the current prediction, accumulated privacy cost, and the next sampling point, in addition to the set of results provided at the end of a task.

3.4 Batch

With prepared datasets, the audience can specify different parameter settings, examine utility results, and perform more analysis and interesting studies. Figure 6 gives two examples of such studies conducted with `linear` data. Users can combine filtering options, i.e. the Kalman filter or particle filter, with fixed rate sampling or adaptive sampling, and compare the utility results. As is shown in Figure 6(a), the performance of fixed-rate sampling method heavily depends on the pre-defined sampling interval; on the other hand, the adaptive sampling method with feedback control (denoted as PID) is comparable to the optimal fixed-rate sampling performance, without *apriori* knowledge.

To compare with the state-of-the-art, FAST also implements the differentially private algorithm based on Discrete Fourier Transform in [10], denoted as DFT. In short, the DFT algorithm transforms the original series with Discrete Fourier Transform, perturbs the coefficients with Laplace noises, reconstructs and releases the inversed series. However, it is only applicable in batch mode. Figure 6(b) illustrates a comparison between FAST algorithms (`KF+PID` and `PF+PID`), DFT, and LPA, with various α values. The audience can observe that FAST achieves lower relative errors than LPA and DFT in most settings, providing higher utility without compromising privacy.

4. CONCLUSION

We have proposed FAST, a tool for monitoring real-time aggregates under differential privacy with filtering and adaptive sampling. The key innovation is that FAST utilizes feedback loops based on observed (perturbed) values to dynamically adjust the filtering model as well as the sampling rate. Our empirical studies across multiple data sets confirm the effectiveness and the superior performance of FAST algorithms with respect to the state-of-the-art methods. The real-time feature and accurate release provided by FAST will facilitate data holders to continuously share private aggregate, thus enabling important data monitoring applications, such as disease surveillance and traffic monitoring. Future work may include utility analysis in order to advise users on the selection of privacy budget and data model parameters, as well as an automatic tuning feature for controller parameters in adaptive sampling.

5. ACKNOWLEDGMENTS

This research was supported in part by NSF grant CNS-1117763, AFOSR grant FA9550-12-1-0240.

6. REFERENCES

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[2] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, 2008.

[3] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284. Springer, 2006.

[4] L. Fan and L. Xiong. Adaptively sharing time-series with differential privacy. *CoRR*, abs/1202.3461, 2012.

[5] L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. CIKM '12, 2012.

[6] S. Garfinkel and M. D. Smith. Guest editors' introduction: Data surveillance. *IEEE Security and Privacy*, 4(6):15–17, Nov. 2006.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

[8] L. Kazemi and C. Shahabi. A privacy-aware framework for participatory sensing. *SIGKDD Explorations*, 13(1):43–51, 2011.

[9] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. SIGMOD '09, 2009.

[10] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, 2010.