Scalable Distributed Data Anonymization

Sabrina De Capitani di Vimercati*, Dario Facchinetti†, Sara Foresti*,
Gianluca Oldani†, Stefano Paraboschi†, Matthew Rossi†, Pierangela Samarati*
* Università degli Studi di Milano, Italy – Email: *firstname.lastname*@unimi.it
† Università degli Studi di Bergamo, Italy – Email: *firstname.lastname*@unibg.it

***Abstract*—We present an approach for enabling a distributed anonymization process over large collections of sensor data. Our approach anonymizes large datasets (which might not fit in main memory) using an arbitrary number of workers within the Spark framework. We describe how to parallelize the anonymization process through a proper partitioning of the dataset. Our experimental evaluation shows that the proposed approach is scalable and do not affect the quality of the anonymized dataset.**

I. Introduction

Almost every object we use in our everyday life already is or is going to be smart, equipped with sensors that constantly collect information about ourselves and the environment where we live (e.g., smart cars monitor the position of the car, engine configuration, tire pressure, etc.). Such data are valuable and may need to be shared with others (e.g., to design better solutions for autonomous driving) without, however, violating the privacy of the individuals to whom they refer.

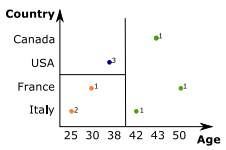
Guaranteeing privacy in datasets containing possible identifying and sensitive information requires not only refraining from publishing explicit identities, but also obfuscating data that can leak (disclose or reduce uncertainty of) such identities as well as their association with sensitive information. k-anonymity [1], [2], extended with ℓ-diversity [3], offers such protection. k-anonymity requires generalizing values of the *quasi-identifier* attributes (i.e., attributes that leak information on respondent's identities exploiting linkage with external sources) to ensure each quasi-identifier combination of values to appear at least k times. ℓ-diversity considers each sensitive attribute in such operation so to ensure each combination of quasi-identifier values to be associated with at least ℓ different values of the sensitive attribute (see Figure 1(c)).

While simple to express, k-anonymity and ℓ-diversity are far from simple to enforce, given the need to balance privacy (in terms of the desired k and ℓ) and utility (in terms of information loss due to generalization). Also, the computation of an optimal solution requires evaluating (based on the dataset content) which quasi-identifying attributes generalize and how, and hence demands complete visibility of the whole dataset for operating the generalization steps. Hence, existing solutions implicitly assume to operate in a centralized environment. Such an assumption clearly does not fit pervasive systems where the amount of data collected is huge (there are widely circulating estimates that a smart car will upload to the cloud 25GB per hour). While scalable distributed architectures can help in performing computation on such large datasets, their use in computing an optimal k-anonymous solution requires careful design. In fact, a simple distribution of the load among workers would affect either the quality of the solution or the scalability of the computation (requiring expensive synchronization and data exchange among workers [4]).

This work was supported in part by the EC within the H2020 Program under projects MOSAICrOWN and MARSAL, by the Italian Ministry of Research within the PRIN program under project HOPE, and by JPMorgan Chase & Co under project "k-anonymity for AR/VR and IoT/5G".

Age	Country	TopSpeed
25	Italy	132
25	Italy	132
30	France	128
42	Italy	110
50	France	115
43	Canada	115
38	USA	126
38	USA	127
38	USA	140

(a)



(b)

Age	Country	TopSpeed
[25-30]	Europe	132
[25-30]	Europe	132
[25-30]	Europe	128
[42-50]	World	110
[42-50]	World	115
[42-50]	World	115
38	USA	126
38	USA	127
38	USA	140

(c)

Fig. 1. An example of a dataset (a), its spatial representation and partitioning (b), and a 3-anonymous and 2-diverse version (c), considering quasi-identifier `Age` and `Country` and sensitive attribute `TopSpeed`

Our contribution. We demonstrate our scalable, efficient, and effective approach for the distributed enforcement of k-anonymity and ℓ-diversity requirements on large datasets. Our solution is based on an adaptation of Mondrian [5], revised to operate without requiring knowledge of the complete dataset. Mondrian is a multi-dimensional algorithm that has established itself as an efficient and effective approach for achieving k-anonymity. Mondrian leverages a spatial representation of the data, mapping each quasi-identifier attribute to a dimension and each combination of values of the quasi-identifier attributes as a point in such a space. Mondrian then recursively cuts the tuples in each partition (the whole dataset at the first step) based on their values (lower/higher than the median) for a quasi-identifying attribute chosen at each cut. The algorithm terminates when any further cut would generate sub-partitions with less than k tuples, at which point values of the quasi-identifier attributes in a partition are substituted with their generalization. Figure 1(b) shows the spatial representation and partitioning of the dataset in Figure 1(a), where the number associated with each data point is the number of tuples with such values for the quasi-identifier in the dataset.

We have extended Mondrian designing a solution for partitioning data for distribution to workers without requiring knowledge of the whole dataset. We have implemented such an approach providing parallel execution on a dynamically chosen

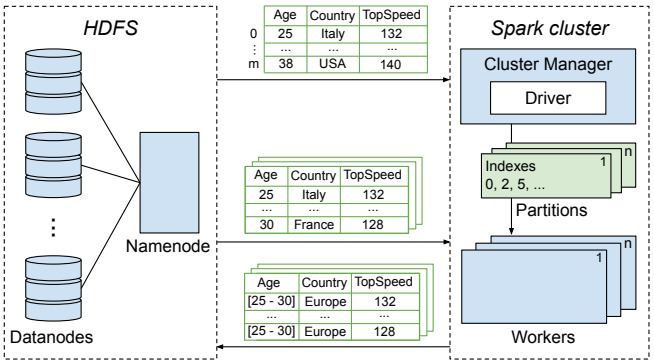


Fig. 2. Architecture and working of the distributed anonymization system

number of workers. The design of our partitioning approach aims at limiting the need for workers to exchange data, by splitting the dataset into as many partitions as the number of workers, which can independently run a revised version of Mondrian on their portion of the data. The experimental evaluation shows that our solution provides scalability, while not affecting the quality of the computed solution.

II. Distributed anonymization

We illustrate the architecture and working of our system (available at https://github.com/mosaicrown/mondrian), supporting the distributed anonymization of large datasets.

A. Architecture

Figure 2 illustrates the architecture of our system, which includes two clusters: an *Hadoop Distributed File System* (HDFS) *cluster*, a well known and widely used solution for data storage and management, and a *Spark cluster* for data processing. Data are split in smaller blocks stored at *datanodes*. A *namenode* in the HDFS cluster manages the data stored at the datanodes and the access requests to them. For data processing, we have opted for *Spark* because it is a widely used engine for big data analytics that is fully compatible with the HDFS cluster. Among the nodes in the Spark cluster, one acts as *Spark Cluster Manager* and coordinates the work of the other nodes in the cluster, acting as *workers*.

Our distributed SPARK anonymization application has been developed in Python to leverage the Pandas framework, which can be conveniently used for managing large data collections. The application is associated with a *Spark Driver*. The Spark Driver, which runs on the Spark Cluster Manager, is responsible for converting the application into a set of jobs that are then divided into smaller execution units, called tasks. The tasks are allocated to workers by the Spark Cluster Manager.

B. Distributed anonymization algorithm

Our application operates in three steps (Figure 2): *pre-processing*, which partitions the dataset and distributes tasks to workers; *anonymization*, which anonymizes the dataset; *wrap-up*, which computes the information loss and collects other information related to the anonymization process.

Pre-processing. The first problem addressed consists in deciding how the dataset can be partitioned by the Spark Driver among the n available workers, in such a way that each worker can independently apply the anonymization algorithm on the portion of data assigned to it, without incurring in too much information loss. We first observe that, while a random partitioning of the dataset would work, it may increase the information loss. We therefore apply a strategy similar to the strategy used by the original Mondrian for creating sub-partitions: we first select an attribute of the quasi-identifier on which to partition the dataset and then create n partitions (one for each worker) depending on the values of the selected attribute. The attribute can be selected by applying different metrics (our tool supports maximum entropy, minimum entropy, and maximum span) that, however, require to have the dataset in main memory to determine the distribution of the quasi-identifying attributes' values. To overcome this problem, we operate on a *sample* of the dataset (whose size is a configuration parameter of our tool) that fits into the main memory of the Spark Driver. Based on the randomly extracted sample, the Spark Driver determines the most suitable attribute, and partitions the tuples in the dataset according to the n-quantiles. We note that, as confirmed by the experimental results (Section III), operating on a sample of tuples for performing the first partitioning of the dataset does not affect the quality of the solution.

Anonymization. The Spark Cluster Manager assigns the task of anonymizing each partition determined in the pre-processing step to a worker, depending on different factors (e.g., the workload, the datanode where data are stored). To make the system scalable, our implementation forces each partition to be assigned to a different worker. Each worker then downloads from the HDFS datanodes its portion of the dataset, and runs a revised version of Mondrian, without the need of interacting with the other workers. Our revised version of Mondrian differs from the original one in two aspects: *1)* the attribute selected for partitioning is determined by applying the same metric used in the pre-processing step; *2)* the partitioning is performed considering both the k-anonymity and ℓ-diversity requirements. When the partitions cannot be further divided without violating k-anonymity nor ℓ-diversity, the tuples in each partitions are generalized. Our tool implements different generalization strategies, suited for different kinds of data (e.g., ranges for numeric attributes, user-defined generalization hierarchies for categorical attributes). Before storing the anonymized portion of dataset back at the datanodes, each worker computes the information loss on its portion of the dataset and sends the result to the Spark Driver (see next step).

Wrap-up. To assess the quality of the anonymized dataset, the Spark Driver computes the information loss produced by our distributed anonymization algorithm. To this end, the Spark Driver combines the values of the information loss received from the workers. Such a combination is done depending on the information loss metric adopted. Our tools support two of the most common metrics, that is, the Discernibility Penalty (DP) and the Global Certainty Penalty (GCP) [6].

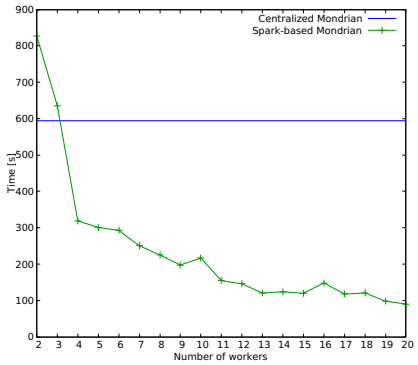


Fig. 3. Execution times of the centralized version and distributed version varying the number of workers

III. Experimental results

To assess the scalability of our approach and its limited impact on information loss, we have tested it over the IPUMS USA dataset [7], which has become a de-facto benchmark for anonymization solutions. The dataset includes 500,000 tuples. We assume the quasi-identifier to include attributes `State FIP Code`, `Age`, `Education Number`, `Occupation`, and the sensitive attribute to be `Income`. We have simulated a distributed environment using a single server through Docker containers. Each node in the architecture in Figure 2 runs in a different Docker container. The server is a 12 cores (24 threads) AMD Ryzen 3900X CPU, with 64 GB RAM and 2 TB SSD, running Ubuntu 20.04 LTS, Apache Spark 3.0.1, Hadoop 3.2.1, and Pandas 1.1.3. The distributed algorithm operates over workers equipped with 2GB of RAM and 1 CPU core each. The centralized algorithm relies on 1 CPU core only, with no limitation on the use of the RAM. Our experiments aim at comparing *1)* the execution time and *2)* the information loss of our distributed approach with those of the centralized version of Mondrian.

Execution time. Figure 3 illustrates the execution time (in seconds) for computing a 3-anonymous and 2-diverse version of the IPUMS USA dataset. The figure shows the execution time of our distributed (Spark-based) Mondrian varying the number of workers between 2 and 20. The execution time of the distributed Mondrian decreases, as expected, when the number of workers grows with a saving with respect to the execution time of the centralized Mondrian that ranges from 46% to 85% when using more than 3 workers. This confirm the scalability of our distributed approach. It is interesting to note that the centralized Mondrian is more efficient than the distributed one when the number of workers is low (2 or 3 in our experiments). This is due to the constant initialization time paid by the distributed implementation for setting distribution and interoperation among workers, and by the different libraries used by the centralized implementation (NumPy) and by the distributed implementation (Spark APIs).

Information loss. We first observe that the information loss caused by distribution can be impacted by: *1)* the number of workers (and hence of partitions), and *2)* the size of the sample used to partition the dataset. Figure 4 illustrates the average information loss (and its variance) obtained in 5 runs of the centralized and distributed (with 5, 10, and 20 workers) Mondrian for computing a 5-anonymous and 2-diverse version of the IPUMS USA dataset, assuming 0.01% and 100% sampling. In the table, 100% sampling corresponds to the centralized Mondrian, since in our experiments the information loss is substantially not affected by distribution.

	100% (centralized)	**0.01% sampling**		
		5 workers	**10 workers**	**20 workers**
DP	$1.24e7$	$1.23e7$ ($\pm 4e5$)	$1.26e7$ ($\pm 4e5$)	$1.33e7$ ($\pm 1e5$)
GCP	6.44	6.47 (± 0.08)	6.49 (± 0.07)	6.46 (± 0.10)

Fig. 4. DP and GCP information loss with 100% and 0.01% sampling

The results we obtained confirm that, as expected, information loss grows with the number of workers (i.e., values in DP and GCP lines in Figure 4 grow when moving from left to right), but the impact is negligible. Also, the results show that sampling has a very limited impact on information loss (i.e., values obtained with 0.01% sampling are slightly higher than the values obtained with 100% sampling). For instance, GCP increases of less than 2% when passing from the centralized version with 100% sampling to the distributed version with 20 workers and 0.01% sampling. DP has a similar trend.

We can then conclude that parallelization provides high scalability at a limited cost in terms of information loss.

IV. Conclusions

We have proposed a distributed version of Mondrian that provides scalability without affecting information loss and leveraging an arbitrary number of independent workers.

Our demo shows the working of our distributed Spark anonymization application. Parameter settings, work distribution, and anonymization results are conveniently controllable via a web interface. The interface enables setting the parameters for privacy (i.e., k and ℓ) and distribution (i.e., number of workers) and provides a visual representation of the system working, as well as of the privacy and utility guarantees for different information loss metrics.

References

[1] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, November/December 2001.

[2] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati, "k-Anonymity," in *Secure Data Management in Decentralized Systems*, T. Yu and S. Jajodia, Eds. Springer-Verlag, 2007.

[3] A. Machanavajjhala, J. Gehrke, and D. Kifer, "ℓ-diversity: Privacy beyond k-anonymity," in *Proc. of ICDE*, Atlanta, GA, USA, April 2006.

[4] F. Ashkouti, K. Khamforoosh, and A. Sheikhahmadi, "DI-Mondrian: Distributed improved Mondrian for satisfaction of the ℓ-diversity privacy model using Apache Spark," *Information Sciences*, vol. 546, pp. 1–24, 2021.

[5] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Proc. of ICDE*, Atlanta, GA, USA, 2006.

[6] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A.-C. Fu, "Utility-based anonymization for privacy preservation with less information loss," *ACM SIGKDD Explorations Newsletter*, vol. 8, no. 2, pp. 21–30, 2006.

[7] S. Ruggles, S. Flood, R. Goeken, J. Grover, E. Meyer, J. Pacas, and M. Sobek, "IPUMS USA: Version 10.0 [dataset]," Minneapolis, MN: IPUMS, 2020, https://doi.org/10.18128/D010.V10.0.