Understanding and Benchmarking the Impact of GDPR on Database Systems

(Experiments and Analysis paper)

Supreeth Shastri
Computer Science
University of Texas at Austin

Vinay Banakar
Hewlett Packard Enterprise

Melissa Wasserman
School of Law
University of Texas at Austin

Arun Kumar
Computer Science
University of California San Diego

Vijay Chidambaram
Computer Science
University of Texas at Austin

ABSTRACT

The General Data Protection Regulation (GDPR) was introduced in Europe to offer new rights and protections to people concerning their personal data. We investigate GDPR from a database systems perspective, translating its legal articles into a set of capabilities and characteristics that compliant systems must support. Our analysis reveals the phenomenon of *metadata explosion*, wherein large quantities of metadata needs to be stored along with the personal data to satisfy the GDPR requirements. Our analysis also helps us identify the new workloads that must be supported under GDPR. We design and implement an open-source benchmark called *GDPRbench* that consists of workloads and metrics needed to understand and assess personal-data processing database systems. To gauge how ready the modern database systems are for GDPR, we modify Redis and PostgreSQL to be GDPR compliant. Our evaluations show that this modification degrades their performance by up to 5×. Our results also demonstrate that the current database systems are two to four orders of magnitude worse in supporting GDPR workloads compared to traditional workloads (such as YCSB), and also do not scale as the volume of personal data increases. We discuss the real-world implications of these findings, and identify research challenges towards making GDPR compliance efficient in production environments. We release all of our software artifacts and datasets at http://www.gdprbench.org

PVLDB Reference Format:
Supreeth Shastri, Vinay Banakar, Melissa Wasserman, Arun Kumar, and Vijay Chidambaram. Understanding and Benchmarking the Impact of GDPR on Database Systems. *PVLDB*, 13(X): xxx-yyy, 2020.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

"Measure what is measurable, and make measurable what is not so."

Galileo Galilei

Proceedings of the VLDB Endowment, Vol. 13, No. X
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

The European Union enacted the General Data Protection Regulation (GDPR) [36] on May 25th 2018, in response to a widespread abuse of personal data. While monetizing personal data at scale has existed since the early dot-com days, a systemic disregard for the privacy and protection of personal data is a recent phenomenon. For example, in 2017, Uber secretly paid off [26] hackers to delete the stolen personal data; Yahoo! confessed [31] that three years ago, a theft had exposed all 3 billion of its user records; Facebook admitted [41] that their APIs allowed illegal harvesting of user data, which in turn influenced the U.S. and U.K. democratic processes.

GDPR rights and responsibilities. To deter such practices, GDPR declares the privacy and protection of personal data as a fundamental right of all European people. It grants several new *rights to the EU consumers* including the right to access, right to rectification, right to be forgotten, right to object, and right to data portability. GDPR also assigns *responsibilities to companies* that collect and process personal data. These include seeking explicit consent before using personal data, notifying data breaches within 72 hours of discovery, maintaining records of processing activities, among others. Failing to comply with GDPR could result in hefty penalties: up to €20M or 4% of global revenue, whichever higher. For instance, in January 2019, Google was fined €50M for lacking customer consent in their ads personalization [37]; in July 2019, British Airways was fined £184M for failing to safeguard personal data of their customers [33].

GDPR compliance is challenging. Compliance with GDPR is challenging for several reasons. First, GDPR's interpretation of personal data is broad as it includes any information that relates to a natural person, even if it did not uniquely identify that person. For example, search terms sent to Google are covered under GDPR. This vastly increases the scope of data that comes under GDPR purview. Second, several GDPR requirements are fundamentally at odds with the design principles and operating practices of modern computing systems [40]. Finally, several GDPR regulations are intentionally vague in their technical specification to accommodate future advancements in technologies. This causes confusion among developers of GDPR-compliant systems. It is no surprise that recent estimates [8, 23] peg the compliance rate to be <50%, even amongst companies that believe they ought to be compliant.

Analyzing GDPR. In this work, we investigate how to understand, achieve, and benchmark GDPR compliance of database systems. To do so, we analyze GDPR and distill its articles into capabilities and characteristics that datastores must support. We make three key observations in our analysis.

1. We identify and characterize the phenomenon of *metadata explosion*, whereby every personal data item is associated with up to seven metadata properties (like its purpose, time-to-live, objections etc) that govern its behavior. By elevating each personal data item into an active entity that has its own set of rules, GDPR mandates that it could no longer be used as a fungible commodity. This is significant from a database standpoint as it severely impacts both the control- and data-path operations of datastores.

2. We observe that GDPR's goal of *data protection by design and by default* conflicts with the traditional system design goals of optimizing for performance, cost, and reliability. For example, in order to investigate and notify any data breaches, GDPR requires a record to be kept of all the interactions with personal data. From a datastore perspective, this turns every read operation into a read followed by a write.

3. Lastly, we identify that GDPR allows new forms of interactions with the datastore. We discuss the characteristics of these *GDPR queries*, and their implications for database systems.

GDPRbench. As our analysis reveals, GDPR significantly affects the design and operation of datastores that hold personal data. However, none of the existing benchmarks recognize the abstraction of personal data: its characteristics, storage restrictions, or interfacing requirements. We design and implement *GDPRbench*, a new open-source benchmark that represents the functionalities of a datastore deployed by a company that collects and processes personal data. The design of GDPRbench is informed by painstaking analysis of the legal cases arising from GDPR from its first year of roll-out. GDPRbench is composed of four core workloads: *Controller*, *Customer*, *Processor*, and *Regulator*; these core workloads are not captured by any database benchmark available today. GDPRbench captures three benchmarking metrics for each workload: correctness, completion time, and space overhead.

Evaluating GDPR-Compliant DBMS. Finally, to gauge how ready the modern database systems are for GDPR, we take two widely-used, open-source database systems, Redis [10] (an in-memory NoSQL store) and PostgreSQL [9] (a relational DBMS), and modify them to be GDPR-compliant. We followed recommendations from the developers of these tools [12, 9] in making them GDPR-compliant; the goal was to make minimal changes, not to redesign the systems for GDPR compliance. While both systems are able to achieve GDPR compliance with a small amount of effort, the resulting systems experienced a performance slow down of 5$\times$ and 2$\times$ respectively. We evaluated the performance of GDPR-compliant Redis and PostgreSQL using GDPRbench. We observed that both systems were operating at a throughput that is two to five orders of magnitude lower than that of the traditional workloads. Our analyses and experiments identify several implications for administering GDPR-compliant database systems in the real world.

Limitations. Our work exclusively focuses on GDPR. While GDPR is a highly visible privacy regulation, several governments [3, 14] are independently working on their own privacy regulations. We acknowledge that some our findings, analyses, and compliance efforts may not generalize to all the privacy laws. Second, our experiments show a significant drop in performance due to achieving GDPR compliance. Though we identify the technical challenges that must be solved to bridge this gap, we have not attempted to solve these challenges. Finally, the design of GDPRbench is guided by several factors including (i) our interpretation of GDPR, (ii) real-world GDPR case studies, and (iii) the two database systems that we investigated. As such, we recognize that the current iteration of GDPRbench is only a snapshot in time, and may need to evolve as more technical and legal use cases emerge.

Summary of contributions. Our work lays the foundation for understanding and benchmarking the impact of GDPR on database systems. In particular, we make the following contributions:

- **GDPR Analysis:** Our work is the one of the first to explore GDPR from a database systems perspective. We analyze the articles of GDPR, both individually and collectively, to distill them into attributes and actions for database systems. In doing so, we (i) observe the phenomenon of metadata explosion, and (ii) identify the new workloads that personal data systems must support.

- **Design and Implementation of GDPRbench:** To enable customers, companies and regulators interpret GDPR compliance in a rigorous and systematic way, we design an open-source GDPR centric benchmark. In GDPRbench, we model the queries and workloads that datastores encounter in the real-world, and develop metrics to succintly represent their behavior. We make all our software artifacts publicly available at http://www.gdprbench.org.

- **Experimental Evaluation:** We discuss our effort into transforming Redis and PostgreSQL to be GDPR-compliant. Our evaluation shows that due to GDPR compliance, Redis experiences a performance degradation of 5$\times$, and PostgreSQL, 2$\times$. Using GDPRbench, we show the completion time and storage space overhead of these compliant systems against real-world GDPR workloads. Finally, we share our insights on deploying compliant systems in production environments, implications of scaling personal data, as well as research challenges towards making GDPR compliance, fast and efficient.

2. BACKGROUND

We begin with a primer on GDPR including its internal structure and its adoption challenges in the real world. Then, we discuss related work to set a context for our contributions.

2.1 GDPR Overview

The European parliament adopted GDPR on April 14th 2016, and made it an enforceable law in all its member states starting May 25th 2018. Its objective is to set ground rules for processing personal data such that its commoditization does not conflict with the rights and protection of the people.

GDPR is written[1] as 99 *articles* that describe its legal requirements, and 173 *recitals* that provide additional context and clarifications to these articles. The articles (henceforth prefixed with $\mathscr{G}$) could be grouped into five broad categories. $\mathscr{G}$1-11 articles layout the definitions and principles of personal data processing; $\mathscr{G}$12-23 establish the rights of the people; then $\mathscr{G}$24-50 mandate the responsibilities of the data controllers and processors; the next 26 describe the role and tasks of supervisory authorities; and the remainder of them cover liabilities, penalties and specific situations. We expand on the three categories that concern systems storing personal data.

[1] even the CS people in our team found it quite readable!

Principles of data processing. GDPR establishes several core principles governing personal data. For example, 𝒢5 requires that data collection be for a specific purpose, be limited to what is necessary, stored only for a well defined duration, and protected against loss and destruction. 𝒢6 defines the lawful basis for processing, while 𝒢7 describes the role of consent.

Rights of data subjects. GDPR grants 12 explicit and excercisable rights to every data subject (a natural person whose personal data is collected). These rights are designed to keep people in loop throughout the lifecycle of their personal data. At the time of collection, people have the right to know the specific purpose and exact duration for which their data would be used (𝒢13, 14). At any point, people can access their data (𝒢15), rectify errors (𝒢16), request erasure (𝒢17), download or port their data to a third-party (𝒢20), object to it being used for certain purposes (𝒢21), and finally, withdraw from automated decision-making (𝒢22).

Responsibilities of data controllers. The third group of articles outline the responsibilities of data controllers (entities that collect and utilize personal data) and data processors (entities that process personal data on behalf of a controller). To clarify, when Netflix runs their recommendation algorithm on Amazon's MapReduce platform, Netflix is the controller and Amazon, the processor. Key responsibilities include designing secure infrastructure (𝒢24, 25), maintaining records of processing (𝒢30), notifying data breaches within 72 hours (𝒢33, 34), analyzing risks prior to processing large amounts of personal data (𝒢35, 36) and controlling the location of data (𝒢44). Additionally, the controllers should create interfaces for people to exercise their GDPR rights.

2.2 GDPR from a Database Perspective

GDPR defines four entities—controller, customer, processor, and regulator—that interact with the data store. Figure–1 shows how three distinct types of data flows between the GDPR entities and data stores. The database that hosts personal data and its associated metadata (purpose, objections etc.,) is the focus of our work. We distinguish it from the other store that contains non-GDPR and derived data as the rules of GDPR do not apply to them.

The controller is responsible for collection and timely deletion of personal data as well as managing its GDPR metadata throughout the lifecycle. The customers interact with the data store to exercise their GDPR rights. The processor uses the stored personal data to generate derived data and intelligence, which in turn powers the controller's businesses and services. Finally, the regulators interact with the datastores to investigate complaints and to ensure that rights and responsibilities are complied with.

Our focus on datastores is motivated by the high proportion of GDPR articles that concern them. From out of the 99 GDPR articles, 31 govern the behavior of data storage systems. In contrast, only 11 describe requirements from compute and network infrastructure. This should not be surprising given that GDPR is more focused on the control-plane aspects of personal data (like collecting, securing, storing, moving, sharing, deleting etc.,) than the actual processing of it.

2.3 GDPR in the Wild

The first year of GDPR has demonstrated both the need for and challenges of a comprehensive privacy law. On one hand, people have been exercising their newfound rights like the ability to download all the personal data that companies have amassed on them [18], and not been shy to report any shortcomings. In fact, the EU data protection board reports [16] 94622 complaints from individuals and organizations in the first 9 months of GDPR.

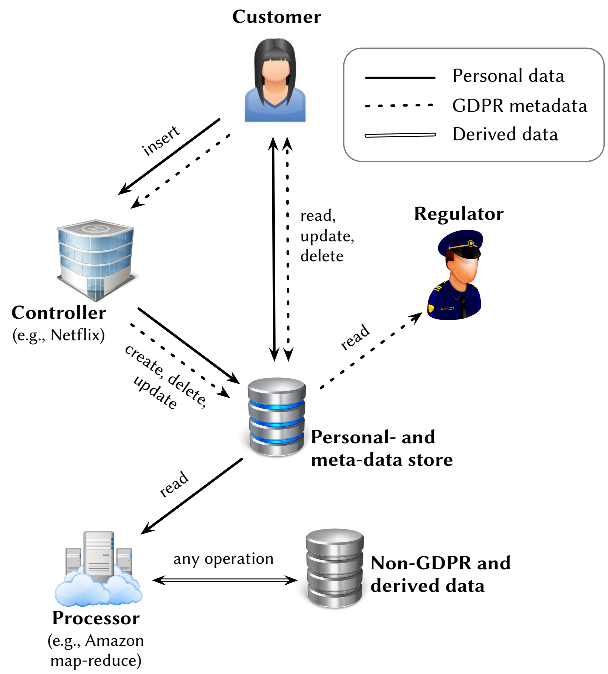


Figure 1: *GDPR defines four roles and distinguishes between three types of data. Only the datastore that contains personal and GDPR metadata comes under the GDPR purview. The arrows out of a datastore indicate read-only access, while the arrows into it modify it. (1) The controller can collect, store, delete and update any personal- and GDPR-metadata, (2) A customer can read, update, or delete any personal data and GDPR-metadata that concerns them, (3) A processor reads personal data and produces derived data, and (4) Regulators access GDPR metadata to investigate customer complaints.*

However, any attempt to regulate decade-long practices of commoditizing personal data is not without consequences. A number of companies like Instapaper, Klout, and Unroll.me voluntarily terminated their services in Europe to avoid the hassles of compliance. Like wise, most of the programmatic ad exchanges [20] of Europe were forced to shut down. This was triggered by Google and Facebook restricting access to their platforms to those ad exchanges that could not verify the legality of the personal data they possessed. But, several organizations could comply by making minor modifications to their business models. For example, media site *USA Today* turned off all advertisements [42], whereas *the New York Times* stopped serving personalized ads [21].

As 𝒢28 precludes employing any data processor that does not comply with GDPR, the cloud providers have been swift in showcasing [49, 48, 35] their compliance. However, given the monetary and technical challenges in redesigning the existing systems, the focus has unfortunately shifted to *reactive security*. It is still an open question if services like Amazon Macie [5], which employs machine learning to automatically discover, monitor, and alert misuse of personal data on behalf of legacy cloud applications would survive the GDPR scrutiny.

Regulators have been active and vigilant as well: the number of ongoing and completed investigations in the first 9 months of GDPR is reported to be 206326. Regulators have already levied penalties on several companies including €50M on Google [37] for lacking a legal basis for their ads personalization, and £184M on British Airways [33] for lacking security of processing. However, the clearest sign of GDPR's effectiveness is in the fact that regulators have received 64684 voluntary data breach notifications from companies in the first nine months of GDPR. In contrast, that number was 945 for the six months prior to GDPR [43].

3. DESIGNING FOR COMPLIANCE

We analyze the articles of GDPR, both individually and collectively, from a database system perspective. The goal of this section is to distill our analysis into attributes and actions that correspond to database systems. We identify three overarching themes: how personal data is to be represented, how personal data is to be protected, and what interfaces need to be designed for personal data. Finally, we determine the impact of these themes on database systems.

3.1 Characterizing Personal Data

GDPR defines personal data to be any information concerning a natural person. As such, it includes both personally identifiable information like credit card numbers as well as information that may not be unique to a person, say search terms sent to Google. This significantly increases the proportion of data that comes under GDPR purview. Also, by not restricting the applicability of GDPR to any specific domains like health and education as in the case of HIPAA [2] and FERPA [1] respectively, GDPR brings in virtually all industries under its foray.

Next, to govern the lifecycle of personal data, GDPR introduces several behavioral characteristics associated with it; we call these *GDPR metadata*. This constitutes a big departure from the evolution of data processing systems, which have typically viewed data as a helper resource that could be fungibly used by software programs to achieve their goals. We discover that, when taken collectively, these metadata attributes convert personal data from an inert entity to *a dynamic entity* that possesses its own purpose, objections, time-to-live etc., such that it can no longer be used as a fungible commodity. Below, we list the seven metadata attributes that must be stored along with every piece of personal data.

1. **Purpose.** $\mathscr{G}$5(1b) states that personal data should only be collected for specific and explicit purposes, and not further processed in a manner incompatible with those purposes. Then, $\mathscr{G}$6(1a) defines processing to be lawful only if the customer has given consent for their data to be used for a specific purpose. As has been established in the recent Google case [37], GDPR explicitly prohibits any purpose bundling.

2. **Time to live.** Given the value of personal data, the longstanding practice in computing has been to preserve them forever (or at least till they are economically viable). However, $\mathscr{G}$5(1e) mandates that personal data shall be kept for no longer than necessary for the purposes for which it was collected. In addition, $\mathscr{G}$13(2a) requires the controller to provide this TTL value to the customer at the time of data collection.

3. **Objections.** $\mathscr{G}$21 grants users a right to object, at any time, to any personal data being used for any purposes. This right is broadly construed, and a controller has to demonstrate compelling legitimate grounds to override it. This property, essentially sets up a blacklist for every personal data item.

4. **Audit trail.** $\mathscr{G}$30 requires controllers and processors to keep the audit trail of all the operations performed on personal data. Then, $\mathscr{G}$33 requires that in the event of a data breach, the controller shall make available a detailed audit trail concerning accesses to personal data.

5. **Origin and sharing.** Every personal data item should have an origin i.e., how it was originally procured, and sharing information i.e., external entities with which it has been shared ($\mathscr{G}$13, 14). The data trail set up by these articles should enable customers to track their personal data in secondary markets.

6. **Automated decision-making.** This concerns the emerging use of algorithmic decision-making. $\mathscr{G}$15(1) grants customers a right to seek information on which of their personal data was used in automated decision-making. Conversely, $\mathscr{G}$22 allows them to request that their personal data be not used for automated decision-making.

7. **Associated person.** $\mathscr{G}$15 enables users to ask for all the personal data that concern them along with all the associated GDPR metadata. As such, it is imperative to store the identification of the person to whom it concerns.

Impact on Database System Design. We call our observation that every personal data item should be associated with a set of GDPR metadata properties as *metadata explosion*. This has significant consequences in both control- and data-path operations of database systems. First, having to store metadata along with the data increases the overall storage space. Second, having to validate each access (for purpose etc.,) and having to update after each access (for audit etc.,), increases the access latency for personal data. Though it may be possible to optimize—for example, by reusing some metadata across records, and caching reads—the overheads cannot be reduced to trivial.

3.2 Protecting Personal Data

GDPR declares (in $\mathscr{G}$24) that those who collect and process personal data are solely responsible for its privacy and protection. Thus, it not only mandates the controllers and processors to proactively implement security measures, but also imposes the burden of proving compliance (in $\mathscr{G}$5(2)) on them. Based on our analysis of GDPR, we identify five security-centric features that must be supported in the database system for it to be compliant.

1. **Timely Deletion.** In addition to $\mathscr{G}$5(1e) that requires every personal data item to have an expiry date, $\mathscr{G}$17 grants customers the right to request erasure of their personal data at any time. Thus, datastores must have mechanisms to allow timely deletion of possibly large amounts of data.

2. **Monitoring and Logging.** As per $\mathscr{G}$30 and $\mathscr{G}$33, the database system needs to maintain audit trails of all operations in both data path (i.e., read or write) and control path (say, changes to access control).

3. **Indexing via Metadata.** Ability to access groups of data based on one or more metadata fields is essential. For example, controllers needing to modify access control ($\mathscr{G}$25(2)) against a given customer's data; $\mathscr{G}$28(3c) allowing processors to access only those personal data for which they have requisite access and valid purpose; $\mathscr{G}$15-18, 20-22 granting customers the right to act on their personal data in a collective manner (deleting, porting, downloading etc.,); finally, $\mathscr{G}$31 allowing regulators to seek access to metadata belonging to affected customers.

4. **Encryption.** $\mathscr{G}$32 requires controllers to implement encryption on personal data, both at rest and in transit. While pseudonymization may help reduce the scope and size of data needing encryption, it is still required of the datastore.

No	GDPR article/clause	What they regulate	Impact on database systems	
			Attributes	Actions
5	Purpose limitation	Collect data for explicit purposes	Purpose	Metadata indexing
5	Storage limitation	Do not store data indefinitely	TTL	Timely deletion
13 14	Information to be provided [...]	Inform customers about all the GDPR metadata associated with their data	Purpose, TTL, Origin, Sharing	Metadata indexing
15	Right of access by users	Allow customers to access all their data	Person id	Metadata indexing
17	Right to be forgotten	Allow customers to erasure their data	TTL	Timely deletion
21	Right to object	Do not use data for any objected reasons	Objections	Metadata indexing
22	Automated individual decision-making	Allow customers to withdraw from fully algorithmic decision-making	Automated decisions	Metadata indexing
25	Data protection by design and default	Safeguard and restrict access to data	—	Access control
28	Processor	Do not grant unlimited access to data	—	Access control
30	Records of processing activity	Audit all operations on personal data	Audit trail	Monitor and log
31	Cooperation w/ supervisory authority	Allow regulators access to system logs	Audit trail	Monitor and log
32	Security of processing	Implement appropriate data security	—	Encryption
33	Notification of personal data breach	Share audit trails from affected systems	Audit trail	Monitor and log

Table 1: *The table maps the requirements of key GDPR articles into database system attributes and actions. This provides a blueprint for designing new database systems as well as retrofitting the current systems into GDPR compliance.*

5. **Access Control.** $\mathscr{G}$25(2) calls on the controller to ensure that by default, personal data are not made accessible to an indefinite number of entities. So, to limit access to personal data based on established purposes, for permitted entities, and for a predefined duration of time, the datastore needs an access control that is fine-grained and dynamic.

Impact on Database System Design. GDPR's goal of *data protection by design and by default* sits at odd with the traditional system design goals of optimizing for cost, performance, and reliability. While our analysis identified a set of just five security features, we note that modern database systems have not evolved to support these features efficiently. Thus, a fully-compliant database system would likely experience significant performance degradation.

3.3 Interfacing with Personal Data

GDPR defines four distinct entities—controller, customer, processor, and regulator—that interface with the database systems (shown in Figure 1). Then, its articles collectively describe the control- and data-path operations that each of these entities are allowed to perform on the database system. We refer to this set of operations as *GDPR queries*, and group them into seven categories:

- **CREATE-RECORD** to allow controllers to insert a record containing personal data with its associated metadata ($\mathscr{G}$24).
- **DELETE-RECORD-BY-**{**KEY**|**PUR**|**TTL**|**USR**} to allow customers to request erasure of a particular record ($\mathscr{G}$17); to allow controllers to delete records corresponding to a completed purpose ($\mathscr{G}$5.1b), to purge expired records ($\mathscr{G}$5.1e), and to clean up all records of a particular customer.
- **READ-DATA-BY-**{**KEY**|**PUR**|**USR**|**OBJ**|**DEC**} to allow processors to access individual data items or those matching a given purpose ($\mathscr{G}$28); to let customers extract all their data ($\mathscr{G}$20); to allow processors to get data that do not object to specific usage ($\mathscr{G}$21.3) or to automated decision-making ($\mathscr{G}$22).
- **READ-METADATA-BY-**{**KEY**|**USR**|**SHR**} to allow customers to find out metadata associated with their data ($\mathscr{G}$15); to assist regulators to perform user-specific investigations, and investigations into third-party sharing ($\mathscr{G}$13.1).
- **UPDATE-DATA-BY-KEY** to allow customers to rectify inaccuracies in personal data ($\mathscr{G}$16).
- **UPDATE-METADATA-BY-**{**KEY**|**PUR**|**USR**} to allow customers to change their objections ($\mathscr{G}$18.1) or alter previous consents ($\mathscr{G}$7.3); to allow processors to register the use of given personal data for automated decision making ($\mathscr{G}$22.3); to enable controllers to update access lists and third-party sharing information for groups of data ($\mathscr{G}$13.3).
- **GET-SYSTEM-**{**LOGS**|**FEATURES**} to enable regulators to investigate system logs based on time ranges ($\mathscr{G}$33, 34), and to identify supported security capabilities ($\mathscr{G}$24,25).

Impact on Database System Design. When taken in totality, GDPR queries may resemble traditional workload, but it would be remiss to ignore two significant differences: (i) there is a heavy skew of metadata-based operations, and (ii) there is a need to enforce restrictions on who could perform which operations under what conditions. These observations make it impractical to store GDPR metadata away from the personal data (say, on backup storage to save money), which in turn may affect system optimizations like caching and prefetching (since the results, and even the ability to execute a query are conditional on several metadata factors).

3.4 Summary

Table–1 summarizes our analysis of GDPR. We identify three significant changes needed to achieve GDPR compliance: ability to handle *metadata explosion*, ability to *protect data by design and by default*, and ability to support *GDPR queries*. While it is clear that these changes will affect the design and operation of all contemporary database systems, we lack systematic approaches to gauge the magnitude of changes required and its associated performance impact. Towards solving these challenges, we design *GDPRbench*, a functional benchmark for GDPR-compliant database systems (in Section-4), and present a case study of retrofitting two popular databases into GDPR compliance (in Section-6).

4. GDPRBENCH

GDPRbench is an open-source benchmark to assess the GDPR compliance of database systems. It aims to provide quantifiable ground truth concerning correctness and performance under GDPR. In the rest of this section, we motivate the need for GDPRbench, and then present its design and implementation.

4.1 Why (a New) Benchmark?

As our analysis in Section-3 reveals, GDPR significantly affects the design and operation of datastores that hold personal data. However, existing benchmarks like TPC and YCSB do not recognize the abstraction of personal data: its characteristics, storage restrictions, or interfacing requirements. This is particularly troublesome given the diversity of stakeholders and their conflicting interests. For example, companies may prefer a *minimal compliance* that avoids legal troubles without incurring much performance overhead or modifications to their systems. On the other hand, customers may want to see a *strict compliance* that prioritizes their privacy rights over technological and business concerns of controllers. Finally, regulators need to arbitrate this customer-controller tussle in a fast-evolving technology world. Thus, having objective means of quantifying GDPR compliance is essential.

A rigorous framework would allow system designers to compare and contrast the GDPR implications of their design choices, as well as enable service providers to better calibrate their offerings. For example, many cloud providers currently report the GDPR compliance of their services in a coarse yes-no format [6], making it difficult for regulators and customers to assess either the compliance levels or performance impact. Finally, many governments around the world are actively drafting privacy regulations. For instance, India's ongoing Personal Data Protection bill [14], and California'a proposed Consumer Privacy Act (CCPA) [3]. This benchmark provides a methodical way to study the efficacy of GDPR regulations, and then adopt suitable parts of this law.

4.2 Benchmark Design

Our approach to benchmark design is guided by two factors: insights from our GDPR analysis, and real-world data from the first year of GDPR roll out. At a high level, GDPRbench models the working of a database deployed by an organization that collects and processes personal data. Below, we describe the key elements of the benchmark design.

4.2.1 Data Records

Given the stringent requirements of GDPR, it is prudent to assume that personal data would be stored separately from other types of data. Thus, our benchmark exclusively focuses on personal data records. Each record takes the form `<Key><Data>` `<Metadata>`, where `<Key>` is a variable length unique identifier, `<Data>` is a variable length personal data, and `<Metadata>` is a sequence of seven attributes, each of which has a three letter attribute name followed by a variable length list of attribute values. We enforce all the fields of the record to have ASCII characters (except for semicolon and comma, which we use as separators). We illustrate this with an example record:

```
ph-1x4b;123-456-7890;PUR=ads,2fa;TTL=365days;
USR=neo;OBJ=∅;DEC=∅;SHR=∅;SRC=first-party;
```

Here, `ph-1x4b` is the unique key and `123-456-7890` is the personal data. Following these two, we have seven attributes namely purpose (PUR), time-to-live (TTL), user (USR), objections (OBJ), automated decisions (DEC), third-party sharing (SHR), and originating source (SRC). Some attributes have a single value, some have a list of values, and a few others have ∅.

4.2.2 Workloads

We define four workloads that correspond to the four core entities of GDPR: controller, customer, processor and regulator. We compose the workloads using the queries outlined in Section-3.3, and concerning only the flow of personal data and its associated metadata (denoted in Figure–1 by thick and dotted lines respectively). Then, we glean over usage patterns and traces from the real-world to accurately calibrate the proportion of these queries and the distribution of the data records they act on. However, since GDPR is just a year old, the availability of said data in the public domain is somewhat limited. Thus, for those situation where no real data is available, we make reasonable assumptions in composing the workloads. The resulting GDPRbench workloads are summerized in Table–2, and described in detail below. While GDPRbench runs these workloads in its default configuration, we make it possible to update or replace them with custom workloads, when necessary.

Controller. The controller workload consists of three categories of operations: (i) creation of records, (ii) timely deletion of records, and (iii) updates to GDPR metadata towards managing access control, categorization, third-party sharing, and location management. While the controller is also responsible for the security and reliability of the underlying storage system, we expect these to be infrequent, non real-time operations and thus, do not include them in our queries.

To determine the frequency and distribution of operations, we rely on three GDPR properties: first, in a steady state, the number of records created must match the number of records deleted (since $\mathcal{G}$5.1 mandates that all personal data must have an expiry date); next, a valid sequence of operation for each record should always be create, updates, and delete in that order; lastly, the controller queries should follow a uniform distribution (since $\mathcal{G}$5.1c prevents the controller from collecting any data that are not necessary or useful). We set the update queries to occur twice as frequently as creates and deletes.

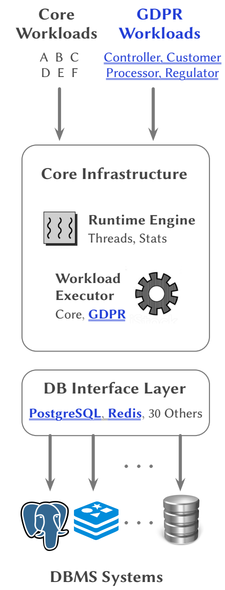
Customer. This workload represents the key rights that customers exercise while interfacing with the database system: (i) the right to delete any of their data, (ii) the right to extract and port all their data, (iii) the right to rectify their personal data, and finally (iv) the right to access and update the metadata associated with a given personal data.

To determine the frequency and distribution of customer queries, we study operational traces from Google's implementation of Right-to-be-Forgotten (RTBF) [15] in Europe. Though GDPR has a namesake article ($\mathcal{G}$17), RTBF is a distinct 2014 EU ruling that allowed individuals to request the search engines to delist URLs that contain inaccurate, irrelevant and excessively personal information from their search results. We gather three high-level takeaways from the Google report: first, they received 2.4 million requests over a span of three years at a relatively stable average of 45k monthly requests. Second, 84.5% of all delisting requests came from individual users. Finally, the requests showed a heavy skew towards a small number of users (top 0.25% users generated 20.8% delisting). Based on these insights, we compose our customer workload by assigning equal weights to all query types and configuring their record selections to follow a Zipf distribution.

Regulator. The role of the regulator is to investigate and enforce the GDPR laws. In case of data breaches or systematic compliance

Workload	Purpose	Operations	Default Weights	Default Distrib.
Controller	Management and administration of personal data	CREATE-RECORD	25%	Uniform
		DELETE-RECORD-BY-{PUR|TTL|USR}	25%	
		UPDATE-METADATA-BY-{PUR|USR|SHR}	50%	
Customer	Exercising GDPR rights	READ-DATA-BY-USR	20%	Zipf
		READ-METADATA-BY-KEY	20%	
		UPDATE-DATA-BY-KEY	20%	
		UPDATE-METADATA-BY-KEY	20%	
		DELETE-RECORD-BY-KEY	20%	
Processor	Processing of personal data	READ-DATA-BY-KEY	80%	Zipf
		READ-DATA-BY-{PUR|OBJ|DEC}	20%	Uniform
Regulator	Investigation and enforcement of GDPR laws	READ-METADATA-BY-USR	46%	Zipf
		GET-SYSTEM-LOGS	31%	
		VERIFY-DELETION	23%	

(a) Core Workloads



(b) Architecture

Figure 2: *GDPRbench core workloads (a), and its architecture (b). The table describes the high-level purpose of each workload along with its composite queries and their default parameters. We select these defaults based on GDPR properties, data from EU regulators, and usage patterns from industry. The architecture diagram shows the components of YCSB that we reused in gray and our GDPR-specific extensions in blue.*

violations, the regulator would summon access to detailed system logs for the period of interest. In case of privacy rights violation of individual customers, they would seek access to the GDPR metadata associated with that particular customer. However, regulators do not need access to any personal data.

To calibrate the regulator workload, we inspect the European Data Protection Board's summary [16] of the first 9 months of GDPR roll out. It reports that the supervisory authorities received 206326 complaints EU-wide. Out of these, 94622 (46%) were direct customer complaints concerning their privacy rights, 64684 (31%) were voluntary data breach notifications from controllers, and the rest (23%) were statutory inquiries against multinational companies, and complaints by non-government and civil rights organizations. We set the weights of regulator queries to match the percentages from this report. Next, in line with the Google RTBF experience, we expect the rights violations and compliance complaints to follow a Zipf distribution.

Processor. The processor, working on behalf of a controller, performs a well-defined set of operations on personal data belonging to that controller. While this role is commonly external, say a cloud provider, the law also allows controllers to be processors for themselves. In either case, the processor workload is restricted to read operations on personal data.

We compose the processor workload to reflect both existing and emerging access patterns. For the former, we refer to the five representative cloud application workloads identified by YCSB, as shown in Table–3. While some operations (like updates and inserts) are not permitted for processors, their access patterns and record distributions are still relevant. For the emerging category, we rely on our GDPR analysis, which identifies access patterns conditioned on metadata attributes like purpose and objection. Since this is still an emerging category, we limit its weight to 20%.

4.2.3 Benchmark Metrics

We identify three metrics that provide a foundational characterization of a database's GDPR compliance: correctness against GDPR workloads, time taken to respond to GDPR queries, and the storage space overhead.

Correctness. We define correctness as the percentage of query responses that match the results expected by the benchmark. This number is computed cumulatively across all the four workloads. It is important to note that correctness as defined by GDPRbench is a *necessary but not sufficient* condition for the database to be GDPR compliant. This is because GDPR compliance includes multitude of issues including data security, breach notification, prior consultation and others that cover the whole lifecycle of personal data. However, the goal of this metric is to provide a basic validation for a database's ability to store and process metadata-based access control.

Completion time. This metric measures the time taken to complete all the GDPR queries, and we report it separately for each workload. For majority of GDPR workloads, completion time is more relevant than the traditional metrics like latency. This is because GDPR operations embody the rights and responsibilities of the involved actors, and thus, their utility is reliant upon completing the operation (and not merely starting them). This is also reflective of the real world, where completion time gets reported more prominently than any other metric. For e.g., Google cloud guarantees that any request to delete a customer's personal data would be completed within 180 days.

Space overhead. It is impossible for a database to comply with the regulations of GDPR without storing large volumes of metadata associated with personal data (a phenomenon described in Section-3.1 as metadata explosion). Since the quantity of metadata overshadows that of personal data, it is an important metric to track. GDPRbench reports this metric as the ratio of total size of the database to the total size of personal data in it. Thus, by definition, it will always be a rational number >1. As a metric, storage space overhead is complementary to completion time since optimizing for one will likely worsen the other. For example, database applications can reduce the storage space overhead by normalizing the metadata. However, this will increase the completion time of GDPR queries by requiring access to multiple tables.

4.3 Implementation

We implement GDPRbench by adapting and extending YCSB. This choice was driven by two factors. First, YCSB is an open-source benchmark with a modular design, making it easy to reuse its components and build new ones on top of it. Second, it is a modern benchmark (released in 2010) and has a widespread community adoption with support for 30+ storage and database systems. In the following, we describe the architecture and operations of GDPRbench.

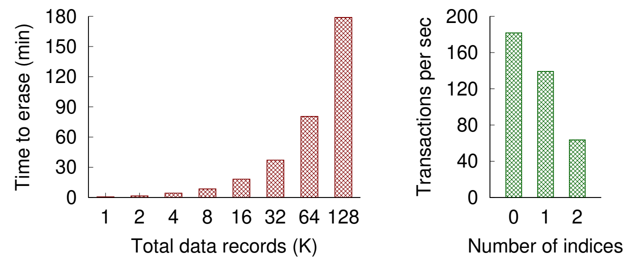
Figure–2b shows the core infrastructure components of YCSB (in gray), and our modifications and extensions (in blue). Alongside the core workloads of YCSB, we create new GDPR workloads that describe operations and their proportions for GDPR roles (as in Table–2). Inside the YCSB core infrastructure, we modify the workload engine to parse GDPR queries and translate them to corresponding storage operations. Note that we reuse the YCSB runtime engine that manages threads and statistics. Finally, the storage interface layer consists of client stubs (one per database system) that translates the generic operations into specific APIs supported by a given storage/database system. Since GDPR introduces many new operations that are not natively supported by most database systems (for example, setting TTL for a record, or operations based on metadata), we had to implement new client stubs for all database systems. Thus far, we have added support for Redis and PostgreSQL, but our goal is to extend this to most major database systems. We have added or modified ∼1300 LoC in the workload engine, and ∼ 400 LoC for Redis and PostgreSQL clients.

5. GDPR-COMPLIANT DBMS

Our goal is to present a case study of retrofitting current generation systems to operate in a GDPR world. Accordingly, we select two widely used open-source systems: Redis [10], an in-memory NoSQL store, and PostgreSQL [9], a fully featured RDBMS. Our effort to transform Redis and PostgreSQL into GDPR compliance is largely guided by the recommendations in their official blogs [12, 9], and other experiences from real-world deployments. While we intend to introduce GDPR compliance into more database systems, and integrate them with GDPRbench, we picked Redis and PostgreSQL as our initial choices as they represent distinct design philosophies, and thus provides a level of generality for our findings. In the following, we describe our code and configuration changes to these two systems, and present microbenchmark measurements.

5.1 Redis

From amongst the features outlined in Section-3, Redis fully supports monitoring; partially supports timely deletion and metadata indexing; but offers no native support for encryption and access control. In lieu of natively extending Redis' limited security model, we incorporate third-party modules for encryption. For data at rest, we use the Linux Unified Key Setup (LUKS) [7], and for data in transit, we set up transport layer security (TLS) using Stunnel [11]. We defer access control to DBMS applications, and in our case, we extend the Redis client in GDPRbench to enforce metadata-based access rights. Next, while Redis offers several mechanisms to generate audit logs, we determine that piggy-



(a) Redis TTL delay (b) Postgres Indices

Figure 3: *Microbenchmarks: (a) Redis' delay in erasing the expired keys beyond their TTL. Our modified TTL algorithm in the GDPR-compliant Redis erases all the expired keys within sub-second latency. (b) PostgreSQL's performance worsens significantly as secondary indices are introduced.*

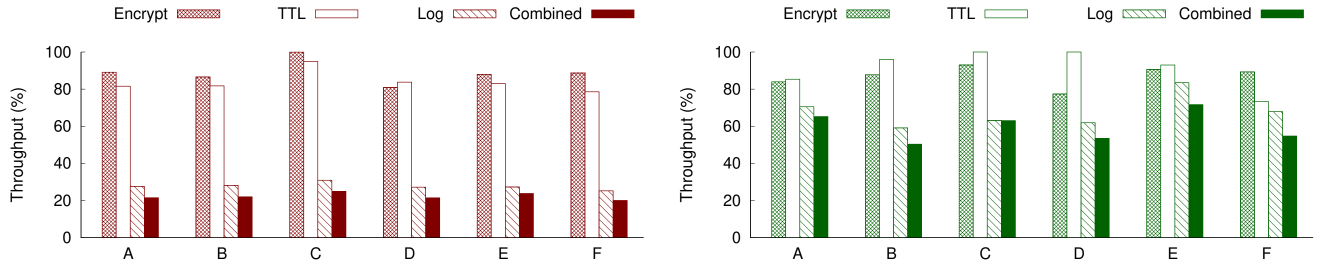
backing on append-only-file (AOF) results in the least overhead. However, since AOF only records the operations that modify Redis' state, we update its internal logic to log all interactions including reads and scans.

Finally, though Redis offers TTL natively, it suffers from indeterminism as it is implemented via a lazy probabilistic algorithm: once every 100ms, it samples 20 random keys from the set of keys with expire flag set; if any of these twenty have expired, they are actively deleted; if less than 5 keys got deleted, then wait till the next iteration, else repeat the loop immediately. Thus, as percentage of keys with associated expire increases, the probability of their timely deletion decreases. To quantify this delay in erasure, we populate Redis with keys having expiry times. The time-to-live values are set up such that 20% of the keys will expire in short-term (5 minutes) and 80% in the long-term (5 days). Figure– 3a then shows the time Redis took to completely erase the short-term keys after 5 minutes have elapsed. As expected, the time to erasure increases with the database size. For example, when there are 128k keys, clean up of expired keys (∼25k of them) took nearly 3 hours. To support a stricter compliance, we modify Redis to iterate through the entire list of keys with associated `EXPIRE`. Then, we re-run the same experiment to verify that all the expired keys are erased within sub-second latency for sizes of up to 1 million keys.

5.2 PostgreSQL

As a feature-rich RDBMS, PostgreSQL offers native support to four of the five GDPR features, with the exception of *Timely deletion.* For encryption, we set up LUKS and SSL (in verify-CA mode). For logging, in addition to the built-in csvlog, we set up a row-level security policy to record query responses. Next, we create metadata indexing via the built-in secondary indices. As with Redis, we enforce metadata-based access control in the external client of GDPRbench. Finally, since PostgreSQL does not offer native support for time-based expiry of rows, we modify the `INSERT` queries to include the expiry timestamp and then implement a daemon that checks for expired rows periodically (currently set to 1 sec).

To efficiently support GDPR queries, an administrator would likely configure secondary indices for GDPR metadata. Interestingly, while PostgreSQL natively supports secondary indices, we observe that its performance begins to drop signficantly when the number of such indices increases as shown in Figure–3b. Using the built-in pgbench tool, we measure throughput on the Y-axis, and the number of secondary indices created on the X-axis. We run this pgbench experiment with a DB size of 15GB, a scale fac-



(a) Redis

(b) PostgreSQL

Figure 4: *Performance overhead of introducing new GDPR features in Redis and PostgreSQL. Our evaluation using YCSB shows that Redis experiences significantly higher ovehead (5×) compared to PostgreSQL (up to 2×).*

tor of 1000, and with 32 clients. Just introducing two secondary indices, for the widely used metadata criteria of *purpose* and *user-id*, reduces PostgreSQL's throughput to 33% of the original.

Key Takeaways. *Introducing GDPR compliance in Redis and PostgreSQL was not an arduous task: Redis needed 120 lines of code changes, and Postgres about 30 lines of scripting. We accomplished all of our modifications, configuration changes, and microbenchmarking in about a person-month. However, as our microbenchmarks on TTL and secondary indices show, even for supported GDPR features, administrators should carefully analyze the impact on their DBMS deployments.*

6. EVALUATION

We evaluate the impact of GDPR on database systems by answering the following questions:

- What is the overhead of GDPR features on traditional workloads? (in Section-6.1)
- How do compliant database systems perform against GDPR workloads? (in Section-6.2)
- How does the scale of personal data impact performance? (in Section-6.3)

Approach. To answer these questions, we use the GDPR compliant versions of Redis and PostgreSQL described in section-5. Next, to quantify the performance overhead of GDPR features on traditional workloads, we use the industry standard Yahoo Cloud Serving Benchmark [19]. Finally, using GDPRbench, we determine the state-of-the-art performance levels of our GDPR-compliant Redis and PostgreSQL against realistic GDPR workloads.

Experimental setup. We perform all our experiments on Chameleon cloud [28]. The database systems are run on a dedicated Dell PowerEdge FC430 with 40-core Intel Xeon 2.3GHz processor, 64 GB RAM, and 400GB Intel DC-series SSD. We choose Redis v5.0 (released March 18, 2019), PostgreSQL v9.5.16 (released Feb 14, 2019), and YCSB 0.15.0 (released Aug 14, 2018) as our reference versions.

6.1 Overheads of Compliance

The goal of this experiment is to quantify the performance overhead of introducing GDPR compliance. To do so, we use the industry standard YCSB [19]. As shown in Table–3, YCSB comprises of 6 workloads that represent different application patterns. For this experiment, we run YCSB with 16 threads; configure it load 2M records and perform 2M operations in each workload category.

Redis. Figure 4a shows the YCSB workloads on the X-axis and Redis' throughput on the Y-axis for each of the newly introduced GDPR security features. We normalize all the values to a baseline version of Redis that has no security. First, we see that encryption reduces the throughput by ∼10%, and our modification towards achieving timely deletion brings it down by ∼20%. Next, we setup Redis to log all its operations via the AoF mechanism (not synchronously in real-time, but in batches synchronized once every second), and see the throughput drops by ∼70%. Finally, when all these features are enabled in tandem, Redis experiences a slowdown of 80%.

Workload		Operation	Application
Load	100%	Insert	Bulk DB insert
A	50/50%	Read/Update	Session store
B	95/5%	Read/Update	Photo tagging
C	100%	Read	User profile cache
D	95/5%	Read/Insert	User status update
E	95/5%	Scan/Insert	Threaded conversation
F	100%	Read-Modify-Write	User activity record

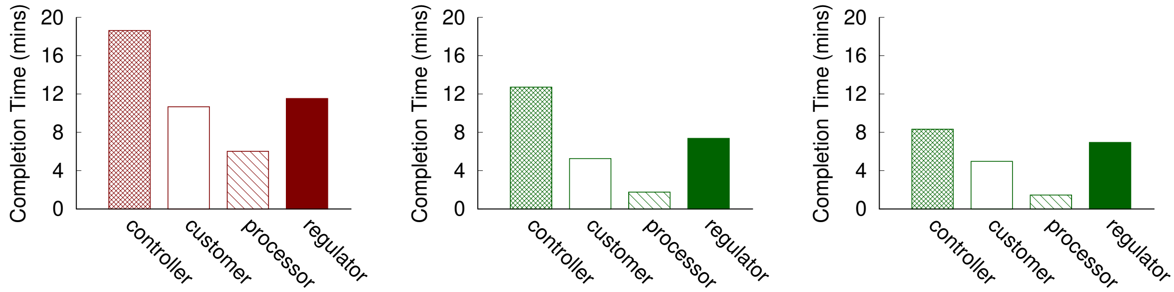
Table 2: *YCSB workload patterns*

PostgreSQL. Similar to Redis, we measure the throughput of PostgreSQL when configured with different security features in Figure 4b. First off, we see that the effect of GDPR on PostgreSQL is not as pronounced as in the case of Redis. This is largely attributable to Redis's minimalist security model as well as the single-threaded design. PostgreSQL experiences 10-20% degradation due to encryption and TTL checks, while logging incurs a 30-40% overhead. When all features are enabled in conjunction, PostgreSQL slows down to 50-60% of its baseline performance.

Summary. *While the performance drop is not surprising in itself since the introduced security measures affect all of the read/write operations, it is the magnitude of the overhead (5× for Redis and ∼2× for PostgreSQL) that makes GDPR compliance debilitating for production environments.*

6.2 GDPR Workloads

While the previous section demonstrated the performance overhead due to GDPR security features, the goal of this section is to quantify how the compliant versions of Redis and PostgreSQL perform against real-world GDPR workloads. To do so, we configure GDPRbench to load 100K personal records, and perform 1K operations for Redis and 10K operations for PostgreSQL on each of its four workloads. We use the default proportion of workload queries and record distributions as specified in Table–2, and run it with 8 threads. Note how the number of operations is two orders



(a) Redis

(b) PostgreSQL

(c) PostgreSQL Metadata-index

Figure 5: *Running GDPRbench workloads on GDPR-compliant Redis and PostgreSQL. Despite performing 10K operations, PostgreSQL is faster than Redis, which is performing only 1K operations for each workload. Enabling metadata indexing in PostgreSQL further reduces its completion time in (c).*

	Personal data size (MB)	Total DB size (MB)	Space factor
Redis	10	35	3.5×
PostgreSQL	10	35	3.5×
PostgreSQL w/ metadata indices	10	59.5	5.95×

Table 3: *Storage space overhead corresponding to Figure-5. In default configuration, GDPRbench has 25 bytes of metadata attributes for a 10 byte personal data record. Our evaluation indicates that introducing secondary indices for all the metadata fields increases the storage space overhead from 3.5× to 5.95×.*

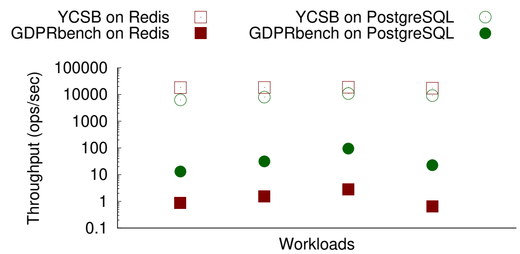


Figure 6: *Representative throughput achieved by Redis and PostgreSQL on YCSB and GDPRbench, under identical conditions. Both systems perform 2-4 orders of magnitude worse for GDPR workloads as opposed to traditional workloads.*

of magnitude lower than in the YCSB configuration. This shows the challenges of supporting GDPR queries on systems not exclusively built for them.

Redis. Figure–5a shows Redis' completion time along Y-axis, and its storage space overhead along Y2-axis. As expected, the processor workload runs the fastest given its heavy skew towards non-metadata based operations. In comparison, all other workloads are 2-3× slower, with the controller workload taking the longest. The figure also benchmarks the metadata explosion. In the default configuation, we see a space overhead ratio of 3.5 i.e., for every byte of personal data inserted, the storage system size increases by 3.5 bytes. Unfortunately, since Redis lacks the support for multiple secondary indices, we do not show any further optimizations.

PostgreSQL. Next, Figure–5b shows the corresponding baseline compliance graph for PostgreSQL. Right away, we see that the completion times are ∼35% faster than Redis for controller and regulator, and ∼50% faster for the other two workloads, despite running a workload with 10× more operations. Our profiling indicates that PostgreSQL, being an RDBMS, is better at supporting complex queries efficiently, on top of implementing GDPR security features with much less overhead (as discussed in section–6.1. However, the storage space overhead remains unchanged compared to Redis.

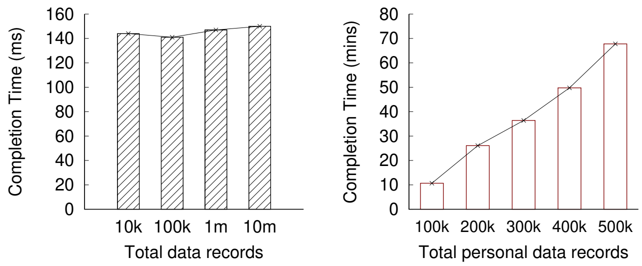
Finally, given the outsized impact of metadata based quries in GDPRbench, we configure PostgreSQL with secondary indices for all metadata. Figure–5c then shows how this improves PostgreSQL's baseline compliance performance. Expectedly, the completion time improves for all the workloads, though the scale of improvement is more pronounced for controller workload. This is primarily because the gain in speed by having secondary indices is annulled to an extent by the overhead of maintaining several secondary indices. However, adding these extra indices increase the storage space overhead from 3.5× to 5.95×.

GDPR vs. traditional workloads. In Figure–6, we compare how Redis and PostgreSQL perform under identical conditions of hardware, software, and configuration against two workloads: YCSB and GDPRbench. For traditional workloads represented by YCSB, both Redis and PostgreSQL achieve throughputs in the range of 10000 operations per second. In contrast, GDPR workloads significantly degrade their performance: by 2-3 orders of magnitude for PostgreSQL, and by 4 orders of magnitude for Redis.

Summary. *GDPRbench reflects the challenges of supporting GDPR specific workloads on retrofitted compliant systems. While both Redis and PostgreSQL suffer from orders of magnitude degradation in their performance compared to traditional workloads, our evaluation shows that feature-rich RDBMS like PostgreSQL performs better than NoSQL stores like Redis.*

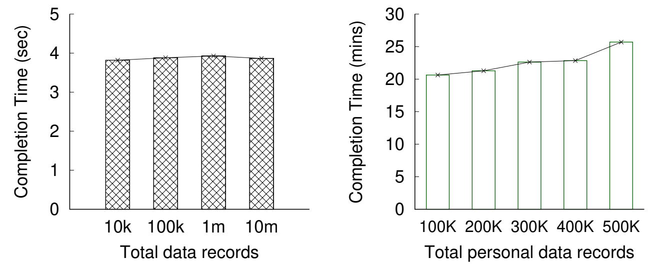
6.3 Effect of Scale

Finally, we explore how an increase in the scale of data affects the systems. In particular, we structure this experiment to reflect a scenario where a company acquires new customers, thus increasing the volume of data in the DB. However, the data of the existing customers remain unchanged. This experiment then measures how Redis and PostgreSQL perform for queries concerning the orignal set of customers. We lay out experiments in two different contexts: first, when the database contains non-personal data, we run YCSB workloads; second, when the database contains per-



(a) YCSB workload-C (b) GDPRbench Customer

Figure 7: *Time taken by Redis to complete 1K operations as the volume of data stored in the DB increases. For the traditional workload in (a), Redis' performance is only governed by the number of operations, and thus remains virtually constant across 3 orders of magnitude change in DB size. However, for GDPR workload in (b), the completion time linearly increases with the DB size.*



(a) YCSB Workload-C (b) GDPRbench Customer

Figure 8: *Time taken by PostgreSQL to complete 10K operations as the DB size scales. Expectedly, PostgreSQL's performance reamins constant for traditional workloads in (a). However, unlike in Redis (Figure-7a), PostgreSQL's GDPR performance worsens only moderately thanks to its use of metadata indices.*

sonal data, we use GDPRbench customer workload. In both cases, we scale the volume of data within the database but perform the same number of operations at every scale. For both GDPR and traditional workloads, we use identical underlying hardware, same version of GDPR-compliant Redis and PostgreSQL software, and retain the same configuration as in section–6.1.

Redis. For Redis, we populate 100K records and perform 1K operations. First, Figure-7a shows Redis' completion time for YCSB workload C. We see that Redis takes almost identical time to complete 1K (read) operations, despite increasing the database volume by 3 orders of magnitude. This is not unexpected as Redis supports efficient, constant-time CRUD operations.

However, when we switch from traditional workloads to GDPR workload, Figure-7b paints a different picture. In this graph, we linearly increase the volume of personal data from 100K to 500K records, and we see a corresponding linear increase in the completion time. This indicates that the completion time is not only a function of the number of operations but also the size of the database. In hindsight, this is not completely unexpected as metadata based queries require O(n) access, especially in absense of secondary indices.

PostgreSQL. Finally, we conduct the same scale experiment on PostgreSQL (metadata-index version) using 100K records and 10K operations. While PostgreSQL's performance for YCSB (shown in Figure-8a) is expectedly similar to that of Redis, its response to GDPR workload (shown in Figure-8b) is much better than that of Redis. While PostgreSQL is still affected by the increase in DB size, the impact on its performance is muted. Our profiling indicates that this is largely due to secondary indices speeding up metadata based queries. But as the DB size increases, the overhead of maintaining multiple secondary indices does introduce some performance degradation.

Summary. *Current generation database systems do not scale well for GDPR workloads. While PostgreSQL with metadata indexing fares better than Redis, neither of them exhibit a scale response that make them production ready, especially in environments with large amounts of personal data.*

7. DISCUSSION

Our experiments and analyses identify several implications for administering GDPR-compliant database systems in the real world, as well as research challenges emerging from it. We discuss them below.

7.1 Real-world Implications

Compliance results in high performance overheads. Our work demonstrates that while it is straight-forward to retrofit Redis and PostgreSQL into GDPR compliance, the resulting performance degradation of 2-5× (in section-6.1) raises fundamental questions of compliance-efficiency tradeoffs. Database engineers and adminstrators should carefully analyze the performance implications of any compliance efforts, especially in production environments. For instance, recommendations from cloud providers such as Amazon Web Services [49], Microsoft Azure [35], and Google Cloud [48] primarily focus on checklist of security features without much attention to their performance implications.

Compliant systems experience challenges at scale. A key takeaway from our scale experiments (in section-6.3) is that naive efforts at achieving GDPR compliance results in poor scalability. Increasing the volume of personal data, even by modest amounts, makes it challenging to respond to customer's GDPR rights in a timely manner, or even to comply with GDPR responsibilities in real-time. Thus, consideration for scale ought to be an important factor in any compliance effort.

Additionally, GDPR quells the notion that personal data, once collected, is largely immutable. In light of GDPR's *right to be forgotten* and *right to rectification*, customers are allowed to exercise much greater control over their personal data. Consequently, traditional solutions to scale problems like replication and sharding would likely incur extra overheads than before. It might be worth investigating the benefits of a GDPR co-processor.

Compliance is easier in RDBMS than NoSQL. We observe that achieving compliance is simpler and effective with RDBMSs than NoSQL stores. In our case, Redis needed two changes at the internal design level as opposed to PostgreSQL, which only needed configuration changes and external scripting. Even from a performance point of view, the drop is steeper in high-performant Redis as compared to PostgreSQL. We hope our findings encourage designers and maintainers of all categories of database systems to reevaluate their design choices, optimization goals, and deployment scenarios in the light of privacy regulations like GDPR.

GDPR is strict in principle yet flexible in practice. Though GDPR is clear in its high-level goals, it is intentionally vague in its technical specifications. Consider $\mathcal{G}$17 that requires controllers to erase personal data upon request by the customer. It does not specify how soon after the request should the data be removed. Let us consider its implications in the real world: Google cloud, which claims GDPR-compliance, describes the deletion of customer data as an iterative process [4] that could take up to 180 DAYS to fully complete.

Such flexibility is not unique to the time of completion. Consider $\mathscr{G}30$ that requires processors to keep an audit trail of interactions with personal data. The regulation does not specify if the log data should be stored in a persistent media, or how often should it be updated, or how easily should it be accessible. Similarly, $\mathscr{G}32$ requires controllers to implement pseudonymization and encryption without specifying any particular algorithms or techniques for either. Thus, having compliance as a spectrum instead of fixed targets, allows database engineers and administrators to explore the tradeoff between strict compliance vs. high performance.

7.2 Research Challenges

Our evaluations show that trivially extending the existing mechanisms and policies to achieve compliance would result in significant performance overheads. We observe two common sources of this: (i) retrofitting new features when they do not align with the core design principles. For example, adding to Redis' minimalist security model, and (ii) using features in ways that are not intended by their designers. For example, enabling continuous logging or multiple secondary indices in production environments. We identify three key challenges that must be addressed to achieve compliance efficiently: *efficient auditing*, *efficient time-based deletion*, and *efficient metadata indexing*.

Another key tussle in the design space is whether to build compliance at the level of individual infrastructure components (i.e., compute servers, and database systems) versus implementing end-to-end compliance of given regulations (i.e., implementing right-of-access in a music streaming service). Both these directions will result in different performance tradeoffs and give rise to different system interfaces. The former approach makes the effort more contained and thus, suits the cloud model better (where GDPR explicitly prohibits selling products and services that do not comply with its regulations). The latter approach provides opportunities for cross-layer optimizations (e.g., avoiding access control in multiple layers).

8. RELATED WORK

A preliminary version of this analysis appeared [39] in a workshop. To the best of our knowledge, this work is one of the first to analyze the impact of GDPR on database systems. While there have been a number of recent work analyzing GDPR from privacy and legal perspectives [34, 25, 47, 13, 44, 17, 45, 22, 27], the database and systems communities are just beginning to get involved. DatumDB [30] proposes an architectural vision for a database that natively supports guaranteed deletion and consent management. Compliance by construction [38] envisions new database abstractions to support privacy rights. In contrast, we focus on the challenges that existing DBMS face in complying with GDPR, and design a benchmark to quantify its impact.

Orthogonal to our focus, researchers are working on implementing and analyzing individual GDPR articles end-to-end. For example, Google researchers [15] have chronicled their experiences implementing the *Right to be Forgotten* for their search service. Two groups of researchers from Oxford University analyzed [24, 46] how GDPR's right to explanation impacts the design of machine learning and artificial intelligence systems. Finally, there is a wealth of blog posts that describe how to achieve GDPR compliance for popular storage systems including MongoDB [29], CockroachDB [32], and Redis [12].

9. CONCLUSION

This work analyzes GDPR from a database systems perspective. We discover the phenomenon of metadata explosion, identify new workloads of GDPR, and design a new benchmark for quantifying GDPR compliance. We find that despite needing to implement a modest number of changes to storage systems, GDPR compliance results in significant performance overheads. Our analyses and experiments identify several implications for administering GDPR-compliant database systems in the real world. We hope that GDPRbench would be useful for customers, controllers, and regulators in interpreting the compliance level of storage systems, and helpful for database designers in understanding the compliance-performance tradeoff.

10. REFERENCES

[1] Family Educational Rights and Privacy Act. *Title 20 of the United States Code, Section 1232g*, Aug 21 1974.

[2] The Health Insurance Portability and Accountability Act. *104th United States Congress Public Law 191*, Aug 21 1996.

[3] California Consumer Privacy Act. *California Civil Code, Section 1798.100*, Jun 28 2018.

[4] Data Deletion on Google Cloud Platform. https://cloud.google.com/security/deletion/, Sep 2018.

[5] Amazon Macie. https://aws.amazon.com/macie/, Accessed Jan 31 2019.

[6] AWS Service Capabilities for Privacy Considerations. https://aws.amazon.com/compliance/data-privacy/service-capabilities/, Accessed Mar 31 2019.

[7] Cryptsetup and LUKS - open-source disk encryption. https://gitlab.com/cryptsetup/cryptsetup, Accessed Jan 2019.

[8] IAPP-EY Annual Privacy Governance Report 2018. Technical report, International Association of Privacy Professionals, Accessed Mar 31 2019.

[9] PostgreSQL: The World's Most Advanced Open Source Relational Database. https://www.postgresql.org/, Accessed Mar 31 2019.

[10] Redis Data Store. https://redis.io, Accessed Jan 2019.

[11] Stunnel. https://www.stunnel.org, Accessed Jan 2019.

[12] C. B. Securing Redis with Redis Enterprise for Compliance Requirements In *Redis Labs Blog*. https://redislabs.com/blog/securing-redis-with-redis-enterprise-for-compliance-requirements/, Jan 10 2018.

[13] D. Basin, S. Debois, and T. Hildebrandt. On Purpose and by Necessity: Compliance under the GDPR. In *Financial Cryptography and Data Security*, 2018.

[14] J. Ben-Avie. India should look to Europe as its model for data privacy. In *Financial Times*. https://www.ft.com/content/56ec37c8-39c0-11e9-9988-28303f70fcff, Mar 4 2019.

[15] T. Bertram, E. Bursztein, S. Caro, H. Chao, R. Feman, P. Fleischer, A. Gustafsson, J. Hemerly, C. Hibbert, and L. Invernizzi. Three years of the Right to be Forgotten. Technical report, Google Inc., Mar 2018.

[16] G. Blyth. Recap: EDPB's first year review of GDPR. In *International Association of Privacy Professionals (IAPP) Privacy Tracker*. https://iapp.org/news/a/recap-edpbs-first-year-review-of-the-gdpr/, Mar 21 2019.

[17] B. Casey, A. Farhangi, and R. Vogl. Rethinking explainable machines: The gdpr's right to explanation debate and the rise of algorithmic audits in enterprise. *Berkeley Technology Law Journal*, 34:143, 2019.

[18] K. Conger. How to Download Your Data With All the Fancy New GDPR Tools. In *Gizmodo*. https://gizmodo.com/how-to-download-your-data-with-all-the-fancy-new-gdpr-t-1826334079, May 25 2018.

[19] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB. In *ACM SoCC*, 2010.

[20] J. Davies. GDPR mayhem: Programmatic ad buying plummets in Europe. In *Digiday*. https://digiday.com/media/gdpr-mayhem-programmatic-ad-buying-plummets-europe/, May 25 2018.

[21] J. Davies. After GDPR, The New York Times cut off ad exchanges in Europe. In *Digiday*. https://digiday.com/media/new-york-times-gdpr-cut-off-ad-exchanges-europe-ad-revenue/, Jan 16 2019.

[22] M. Degeling, C. Utz, C. Lentzsch, H. Hosseini, F. Schaub, and T. Holz. We value your privacy... now take some cookies: Measuring the gdpr's impact on web privacy. In *NDSS*, 2019.

[23] A. A. Forni and R. van der Meulen. Organizations are unprepared for the 2018 European Data Protection Regulation. In *Gartner*, May 2017.

[24] B. Goodman and S. Flaxman. European Union Regulations on Algorithmic Decision-Making and a Right to Explanation. *AAAI AI Magazine*, 38(3), 2017.

[25] S. Greengard. Weighing the impact of gdpr. *Communications of the ACM*, 61(11):16–18, 2018.

[26] M. Isaac, K. Benner, and S. Frenkel. Uber Hid 2016 Breach, Paying Hackers to Delete Stolen Data. In *The New York Times*. https://www.nytimes.com/2017/11/21/technology/uber-hack.html, Nov 21 2017.

[27] F. Kammueller. Formal modeling and analysis of data protection for gdpr compliance of iot healthcare systems. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018.

[28] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth. Chameleon: a scalable production testbed for computer science research. In *Contemporary High Performance Computing: From Petascale toward Exascale*, volume 3 of *Chapman and Hall/CRC Computational Science*, chapter 5. CRC Press, 1 edition, 2018.

[29] M. Keep. GDPR: Impact to Your Data Management Landscape In *MongoDB Blog*. https://www.mongodb.com/blog/post/gdpr-impact-to-your-data-management-landscape-part-1, Aug 29 2017.

[30] T. Kraska, M. Stonebraker, M. Brodie, S. Servan-Schreiber, and D. Weitzner. Datumdb: A data protection database proposal. In *Poly'19 co-located at VLDB 2019*, 2019.

[31] S. Larson. Every single Yahoo! account was hacked - 3 billion in all. In *CNN Business*, Oct 4 2017.

[32] S. Loiselle. What does GDPR compliance mean for my database? In *Cockroach Labs Blog*. https://www.cockroachlabs.com/blog/gdpr-compliance-for-my-database/, July 10 2018.

[33] I. Lunden. UK's ICO fines British Airways a record 183M over GDPR breach that leaked data from 500000 users In *TechCrunch*. https://techcrunch.com/2019/07/08/uks-ico-fines-british-airways-a-record-183m-over-gdpr-breach-that-leaked-data-from-500000-users/, July 8th 2019.

[34] J. Mohan, M. Wasserman, and V. Chidambaram. Analyzing gdpr compliance through the lens of privacy policy. In *Poly'19 co-located at VLDB 2019*, 2019.

[35] A. Rayani. Safeguard individual privacy rights under GDPR with the Microsoft intelligent cloud. In *Microsoft 365 Blog*, May 25 2018.

[36] G. D. P. Regulation. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union*, 59(1-88), 2016.

[37] A. Satariano. Google is fined $57 Million Under Europe's Data Privacy Law. In *The New York Times*. https://www.nytimes.com/2019/01/21/technology/google-europe-gdpr-fine.html, January 21st 2019.

[38] M. Schwarzkopf, E. Kohler, F. Kaashoek, and R. Morris. Gdpr compliance by construction. In *Poly'19 co-located at VLDB 2019*, 2019.

[39] A. Shah, V. Banakar, S. Shastri, M. Wasserman, and V. Chidambaram. Analyzing the Impact of GDPR on Storage Systems. In *USENIX HotStorage*, 2019.

[40] S. Shastri, M. Wasserman, and V. Chidambaram. The Seven Sins of Personal-Data Processing Systems under GDPR. In *USENIX HotCloud*, 2019.

[41] O. Solon. Facebook says Cambridge Analytica may have gained 37M more users' data. In *The Guardian*, Apr 4 2018.

[42] E. Sweeney. Many publishers' EU sites are faster and ad-free under GDPR. In *Marketing Dive*. https://www.marketingdive.com/news/study-many-publishers-eu-sites-are-faster-and-ad-free-under-gdpr/524844/, Jun 4 2018.

[43] E. Targett. 6 Months, 945 Data Breaches, 4.5 Billion Records. In *Computer Business Review*. https://www.cbronline.com/news/global-data-breaches-2018, Oct 9 2018.

[44] W. Tesfay, P. Hofmann, T. Nakamura, S. Kiyomoto, and J. Serna. I read but don't agree: Privacy policy benchmarking using machine learning and the eu gdpr. In *Companion Proceedings of the The Web Conference 2018*, pages 163–166, 2018.

[45] C. Utz, M. Degeling, S. Fahl, F. Schaub, and T. Holz. (un) informed consent: Studying gdpr consent notices in the field. In *ACM CCS*, 2019.

[46] S. Wachter, B. Mittelstadt, and L. Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2), 2017.

[47] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr.(2017). *Harvard Journal of Law & Technology*, 31:841, 2017.

[48] G. C. Whitepaper. Google Cloud and the GDPR. Technical report, Google Inc., May 2018.

[49] C. Woolf. All AWS Services GDPR ready. In *AWS Security Blog*. https://aws.amazon.com/blogs/security/all-aws-services-gdpr-ready/, Mar 26 2018.