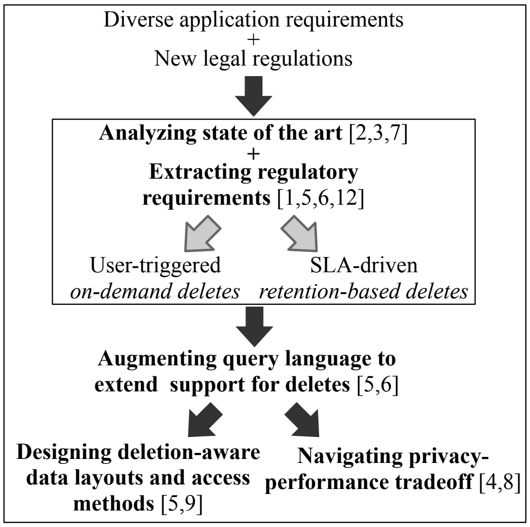
Research Statement

Subhadeep Sarkar (subhadeepsarkar.bitbucket.io)

My research aims at *building data systems for emerging data and applications characteristics* with a focus on *privacy protection of user data from a systems perspective*. Today, businesses, technologies, and scientific applications generate data at unprecedented rates and volumes. Thus, one of the key design goals of modern data systems has been to make them more ingestion-friendly. At the same time, with the emergence of new data protection regulations across the globe, protecting user data privacy has become critical. However, the design principles of write-optimized data systems are fundamentally conflicting with those of a system that protects data privacy. To this end, the goal of my research is to design data systems that can offer high performance in the face of the ever-evolving workloads, while protecting data privacy.

The quest of building ingestion-optimized data systems has led to the birth and widespread adoption of a new class of systems that are based on the *out-of-place paradigm*. By design, out-of-place systems realize updates and deletes *logically* by inserting a newer version of the entries. The logically invalidated entries are retained in the database for arbitrarily long. This has severe implications on both privacy and performance fronts because (i) logical deletion does not guarantee timely and persistent deletion of data, which is a key requirement for several privacy regulations, and (ii) any attempt to demonstrate regulation compliance by periodically purging the data under deletion is extremely costly and severely hurts performance.

Building Deletion-Compliant Data Systems


Diverse application requirements
+
New legal regulations
Analyzing state of the art [2,3,7]
+
Extracting regulatory requirements [1,5,6,12]
User-triggered *on-demand deletes*
SLA-driven *retention-based deletes*
Augmenting query language to extend support for deletes [5,6]
Designing deletion-aware data layouts and access methods [5,9]
Navigating privacy-performance tradeoff [4,8]

My research enables *privacy-through-deletion as a design element* in modern data systems by empowering them with the ability to delete data timely and persistently, without hurting performance. I begin by looking at the core of data systems, i.e., the data structures and access methods to formalize the design space of data systems by introducing the first-order primitives of data layout construction and reorganization techniques [1, 3, 4, 8]. Next, I introduce a workload-aware model to navigate this design space, highlighting how the fundamental elements of data structure and access method design interact with system performance [2, 7, 12]. Based on this, I introduce the tools at the systems and application levels to ensure privacy through deletion [5, 6, 9].

1 Present and Past Research

Out-of-Place Data Structures and LSM-Trees. The ever-increasing need for fast data ingestion and fast query processing have driven the widespread adoption of the out-of-place paradigm across modern NoSQL data systems. The key design principles of these out-of-place, ingestion-optimized storage engines are: (i) realize updates and deletes out of place, without eagerly modifying the target data objects, (ii) lazily merge the new entries with the older ones, and (iii) maintain order among the ingested entries to ensure query correctness. The log-structured merge (LSM) paradigm fits the bill perfectly, and due to this, LSM-trees are widely used as the out-of-place, storage-based data structure in several NoSQL data stores. LSM-trees store data as key-value pairs and arrange them as a hierarchical collection of sorted components or levels at the device level [3].

Deletes in LSMs. As an out-of-place data structure, instead of deleting a target entry in-place, LSM-trees insert a special type of key-value pair, known as a *tombstone*, that logically invalidates the target entries [9]. Logical data deletion is a quintessential LSM operation, but it does not guarantee purging of the data under deletion within a definite time-frame. While the data is marked as invalid essentially remains inaccessible to external users, in practice, the logically deleted entries are retained for arbitrarily long in the system [9]. In fact, most out-of-place data stores are built with the underlying assumption of perpetual data retention in order to gain more insights from the user and organizational data, hence *timely persistent deletion* has not been part of their design goals.

The Legal Frontier & the Overarching Question. In recent years, a number of government-driven efforts across the globe unfolded, aiming to protect the privacy of user data and give back to the users the control of their personal data. On the legal side, regulations such as the EU's GDPR, California's privacy protection acts – CCPA and CPRA, and Virginia's VCDPA have been introduced, which mandate data companies to ensure privacy through deletion. GDPR's right to be forgotten, CCPA and CPRA's right to delete, and the deletion right in VCDPA particularly focus on persistent deletion of user data on-demand and in a timely manner. Thus, to demonstrate regulation compliance, service providers must persistently delete all copies of the data under deletion in a time-bound manner. However, as we pointed out earlier, state-of-the-art LSM engines are unable to provide any latency guarantees on persistent deletion of data. In fact, in principle, the design goals of out-of-place systems are antithetical to those of ensuring timely delete persistence [5]. Thus, in practice, service providers end up performing a database-wide operation on a periodic basis (typically, every few weeks) to check and purge all invalid entries that are older than a threshold. This is a remarkably expensive operation that also adversely affects the overall performance of the system. So in a nutshell, modern out-of-place data stores **are unable to support timely data deletion**, and any **attempt to demonstrate compliance with the deletion regulations does not scale**. This begs the question: *how can we make data systems delete data persistently and timely without compromising with the system performance*?

Step 1: Understanding the Design Principles of Storage Engines. Our first work aims to dissect the internal design and operating principles of storage engines at the fundamental level of data structures and access methods. We study the design principles of more than ten state-of-the-art commercial and academic data systems, and then implement them on an open-source storage engine (RocksDB) for fair evaluation and comparison. We then perform upward of 2,000 experiments with various configuration, tuning, and workloads to capture the implication of the different first-order design choices on performance. We uncovered several interesting and seemingly non-intuitive takeaways from this experimental study that would go on to help us building systems that can support timely data deletion by design [8]. Further, to visualize how the design choices affect the performance space, we designed an interactive demonstration website [4].

Step 2: Identifying and Translating the Regulatory Requirements. In parallel to Step 1, we also strived to extract the regulatory requirements for deletes by inspecting the legal policies [12]. We identify the two types of data deletion requests as that must be supported in order to demonstrate compliance [5]. (A) *On-demand deletion*, where the user submits an ad-hoc deletion request, and the service providers must persistently delete the data within a threshold duration. (B) *Retention-driven deletes*, where the user sets a retention duration, and any data older than that, must be persistently deleted from the data stores. Next, we needed to translate these deletion requirements into system-interpretable instructions. The interface of data stores is typically declarative query languages (e.g., SQL, GraphQL, DMX, LINQ, and N1QL) that support expressing complex queries as well as inserting new data, updates, and deletes. Thus, we extended the query language to capture the deletion requirements and express then to the underlying storage engine [6].

Step 3: Supporting On-Demand Deletes Timely and Efficiently. From our experimental analysis in Step, we identified that a databases internal data reorganization/consolidation mechanisms (i.e., compaction for LSM-trees, node split/merge for B-trees, etc.) play a critical role in our quest to persisting logical deletes in a timely and efficient manner. We observed that design decisions such as *when a database reorganizes/consolidates data* and *how exactly the consolidation takes place* influence the entire performance space of a storage engine, including an engine's ability to persist deletes in a timely manner. The key intuitions behind our solution are: (i) use the file-level metadata to track the age of a tombstone, (ii) trigger inter-level data consolidation based on the age of tombstone and a temporal threshold set for a level, and (iii) choose the appropriate files for consolidation, aiming to timely purging of the tombstones. The proposed solution not only guarantees timely persistence of logical deletes, but by do so, also achieves $2\times$-$9\times$ lower space amplification and $1.2\times$-$1.4\times$ higher read performance [9]. The solution is simple and easily integrable to production systems, and a variant of this solution is now supported by RocksDB, a widely used, open-source key-value store.

Step 4: Supporting Retention-Based Deletes. Retention-based deletes entail deletion of a fraction of a database every day on a rolling basis, and state-of-the-art key-value stores are unable to support this operation when data is not already ordered on some time-attribute (which is rare in non-time-series databases). This is a very complex operation as the data is typically stored as sorted on a primary attribute (let's call this the *sort key*), and the deletion is to be performed on some secondary temporal attribute (we call this the *delete key*). Our key intuition to solve this problem was to embed an ordering of the data based on the delete key without completely destroying the order on the sort key. We introduced a new weaved data layout between the (primary) sort attribute and the (secondary) delete attribute. The benefit of this weaved data layout is that for retention-driven deletes, we can discard entire blocks of data at a time, signaling the file system to reclaim this page; essentially converting the delete action to a page reclamation action that has very low latency compared to a full database consolidation. The proposed interweaved data layout reduces the cost of retention-based deletes significantly, between $4\times$ and $31\times$ even for a workload with a relatively small fraction (2%) of deletes [9].

First Steps Toward a Longer-Term Goal. Building data systems that are designed to process deletes gracefully and are able to demonstrate regulation compliance is an eagerly sought after endeavor that benefits multiple stakeholders in the data-driven service framework. In practice, all data systems exhibit an intrinsic trade-off between performance and privacy protection. The high-level challenge is to be able to propel this trade-off curve closer to the Pareto optimal and efficiently navigate it based on workload and target performance. To this end, my current research helps lay out the fundamental principles and the framework, but moving forward, there are still several steps to be taken before we can realize the goal of building truly deletion-compliant data systems.

Psat Research. As part of my doctoral research, I have worked on deploying cloud infrastructure for practical application scenarios, e.g., healthcare and disaster management scenarios and benchmarking fog/edge computing platforms. The results of my doctoral research have been published in several top-tier journals [10, 14, 15, 16, 17, 18, 20, 21, 24] and conferences [19, 22, 23, 25]. My doctoral thesis won the second best PhD thesis award at the 5^{th} IDRBT Doctoral Colloquium [13]. I have also co-authored a book on fog computing and Internet-of-things that was published by CRC Press [11].

2 Future Research

My long term research goal is to design data systems that can meet the diverse application requirements and performance goals in the face of ever-evolving workloads while protecting user data privacy. Toward this, I have identified three key research directions: (i) *designing scalable data systems that optimize energy consumption without hurting performance, thereby, moving toward "greener" computing frameworks*, (ii) *designing data systems tailored to the underlying properties of evolutionary hardware to offer optimal performance*, and (ii) *enabling timely and persistent data deletion for geo-distributed cloud applications*. Given my expertise in designing privacy-aware data systems [1, 3, 5, 6, 9, 12], navigating the performance tradeoffs in modern data systems [2, 7, 4, 8], and building systems on cloud [11, 16, 14, 17, 24], I am well-equipped to address these research challenges.

Green Storage Engines to Manage Exabyte-Scale Data. Data stores will soon be managing exabytes of data that entails unforeseen challenges in terms of infrastructure, cost, and practicality. Storing and processing high volumes of data entails consumption of a remarkably high amount of energy across the several components of the data management pipeline. My goal is to *introduce energy as a design dimension of data structures and data systems*. I plan to redesign the data structures and access methods by reasoning about every single design choice and how it impacts to hardware usage and energy consumption. My outlook is to first create the building blocks by constructing the energy profile of data structures (the smallest unit of systems), and then, recursively create and reason about energy profiles of algorithms, models, systems, and finally, end-to-end pipelines.

Making the Most of Evolving Hardware. As technology advances, evolution of hardware becomes a natural process where hardware becomes more sophisticated in terms of speed, configurability, size, and design of control circuitry. Thus, it is crucial to redesign software so that systems can harness the synergy of hardware-software co-design. My goal is to *revisit and redesign existing data structures and algorithms to take advantage of the underlying hardware*. To give a concrete example, when running LSM-trees on non-volatile memory (NVM), the device and and the data structure both garbage collect in an asynchronous manner. This leads to a multiplicative write amplification cost hurting performance and the device lifetime. To this end, I plan to program the device controller and design algorithms that combines the data structure's garbage collection routine with that of the device to reduce amplification.

Supporting Timely and Persistent Data Deletion in Cloud. With an increasing number of storage engines moving to cloud, deleting user data timely and persistently becomes a very complex problem. Within cloud, data is typically replicated several times and possibly across multiple regions. Deletes and updates to a particular instance of data is propagated lazily to all its replicas. This is non conducive to facilitating timely data deletion, and any attempt to demonstrate compliance with the delete regulation would result into unnecessary data movement and increased operational cost. Further, as replicas may be geo-distributed, they might also be subject to different privacy regulations. Managing data governed by different privacy regulations from the same data store introduces provenance challenges. I propose to address these challenges by *enabling cross-domain data tracking, and domain-specific modular privacy solutions within the cloud instances*. The goal is to design tools and infrastructure that can scale with data, application complexity, as well as tenancy.

References

[1] **Subhadeep Sarkar**, Niv Dayan, and Manos Athanassoulis. *The LSM Design Space and its Read Optimizations*. In Proceedings of the IEEE International Conference on Data Engineering **(ICDE)**, 2023.

[2] Aneesh Raman, **Subhadeep Sarkar**, Matthaios Olma, and Manos Athanassoulis. *Indexing for Near-Sorted Data*. In Proceedings of the IEEE International Conference on Data Engineering **(ICDE)**, 2023.

[3] **Subhadeep Sarkar** and Manos Athanassoulis. *Dissecting, Designing, and Optimizing LSM-Based Data Stores*. In Proceedings of the **ACM SIGMOD** International Conference on Management of Data, pages 12489–2497, 2022.

[4] **Subhadeep Sarkar**, Kaijei Chen, Zichen Zhu, and Manos Athanassoulis. *Compactionary: A Dictionary for LSM Compactions*. In Proceedings of the **ACM SIGMOD** International Conference on Management of Data, pages 2429–2432, 2022.

[5] Manos Athanassoulis, **Subhadeep Sarkar**, Tarikul I. Papon, Zichen Zhu, and Dimitris Staratzis. *Building Deletion-Compliant Data Systems*. **IEEE Data Engineering Bulletin**, 45(1), pages 21–36, 2022.

[6] **Subhadeep Sarkar** and Manos Athanassoulis. *Query Language Support for Timely Data Deletion*. In Proceedings of the International Conference on Extending Database Technology **(EDBT)**, pages 429–434, 2022.

[7] Aneesh Raman, Konstantinos Karatsenidis, **Subhadeep Sarkar**, Matthaios Olma, and Manos Athanassoulis. *BoDS: A Benchmark on Data Sortedness*. In Proceedings of the TPC Technology Conference on Performance Evaluation & Benchmarking **(TPCTC)**, 2022.

[8] **Subhadeep Sarkar**, Dimitris Staratzis, Zichen Zhu, and Manos Athanassoulis. *Constructing and Analyzing the LSM Compaction Design Space*. In Proceedings of the Very Large Databases Endowment **(PVLDB)**, 14(11), pages 2216–2229, 2021.

[9] **Subhadeep Sarkar**, Tarikul I. Papon, Dimitris Staratzis, and Manos Athanassoulis. *Lethe: A Tunable Delete-Aware LSM Engine*. In Proceedings of the **ACM SIGMOD** International Conference on Management of Data, pages 893–908, 2020.

[10] Sudip Misra, Pradyumna K. Bishoyi, and **Subhadeep Sarkar**. *i-MAC: In-Body Sensor MAC in Wireless Body Area Networks for Healthcare IoT*. **IEEE Systems Journal**, 15(3), pages 4413–4420, 2020.

[11] Sudip Misra, **Subhadeep Sarkar**, and Subarna Chatterjee. *Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things*. **CRC Press**, 2019.

[12] **Subhadeep Sarkar**, Jean-Pierre Banâtre, Louis Rilling, and Christine Morin. *Towards Enforcement of the EU GDPR: Enabling Data Erasure*. In Proceedings of the IEEE International Conference of Internet of Things **(iThings)**, pages 1–8, 2018.

[13] **Subhadeep Sarkar**. *Analysis of Delay in Wireless Communication for Healthcare Systems*. **PhD Thesis**, Indian Institute of Technology Kharagpur, 2017.

[14] **Subhadeep Sarkar**, Subarna Chatterjee, Sudip Misra, and Rajesh Kudupudi. *Privacy-Aware Blind Cloud Framework for Advanced Healthcare*. **IEEE Communications Letters,** 21(11), pages 2492–2495, 2017.

[15] Prasenjit Bhavathankar, **Subhadeep Sarkar**, and Sudip Misra. *Optimal Decision Rule-Based Ex-Ante Frequency Hopping for Jamming Avoidance in Wireless Sensor Networks*. **Computer Networks, Elsevier**, 128, pages 172–185, 2017.

[16] **Subhadeep Sarkar**, Subarna Chatterjee, and Sudip Misra. *Assessment of the Suitability of Fog Computing in the Context of Internet of Things*. **IEEE Transactions on Cloud Computing** 6(1), pages 46–59, 2016.

[17] **Subhadeep Sarkar** and Sudip Misra. *Theoretical Modelling of Fog Computing: A Green Computing Paradigm to support IoT Applications*. **IET Networks,** 5(2), pages 23–29, 2016.

[18] **Subhadeep Sarkar** and Sudip Misra. *From Micro to Nano: The Evolution of Wireless Sensor-Based Health Care*. **IEEE Pulse,** 7(1), pages 21–25, 2016.

[19] **Subhadeep Sarkar**, Sudip Misra, and Mohammad S. Obaidat. *Resource Allocation for Wireless Body Area Networks in Presence of Selfish Agents*. In Proceedings of the IEEE Global Communications Conference **(GLOBECOM)**, pages 1–6, 2016.

[20] Sudip Misra and **Subhadeep Sarkar**. *Priority-Based Time-Slot Allocation in Wireless Body Area Networks During Medical Emergency Situations: An Evolutionary Game-Theoretic Perspective*. IEEE Journal of Biomedical and Health Informatics **(JBHI)**, 19(2), pages 541–548, 2015.

[21] **Subhadeep Sarkar**, Sudip Misra, Bandyopadhyay, B., Chandan Chakraborty, and Mohammad S. Obaidat. *Performance Analysis of IEEE 802.15.6 MAC Protocol under Non-Ideal Channel Conditions and Saturated Traffic Regime*. **IEEE Transactions on Computers**, 64(10), pages 2912–2925, 2015.

[22] Subarna Chatterjee, **Subhadeep Sarkar**, and Sudip Misra. *Quantification of Node Misbehavior in Wireless Sensor Networks: A Social Choice-Based Approach*. In Proceedings of the IEEE International Conference on Communication Workshop, **(ICCW)**, pages 1479–1484, 2015.

[23] Subarna Chatterjee, **Subhadeep Sarkar**, and Sudip Misra. *Energy-Efficient Data Transmission in Sensor-Cloud*. International Conference on Applications and Innovations in Mobile Computing, **(AIMoC)**, pages 68-73, 2015.

[24] **Subhadeep Sarkar**, Subarna Chatterjee, and Sudip Misra. *Evacuation and Emergency Management Using a Federated Cloud*. **IEEE Cloud Computing**, 1(4), pages 68–76, 2014.

[25] **Subhadeep Sarkar**, Sudip Misra, Chandan Chakraborty, and Mohammad S. Obaidat. *Analysis of Reliability and Throughput under Saturation Condition of IEEE 802.15.6 CSMA/CA for Wireless Body Area Networks*. In IEEE Global Communications Conference **(GLOBECOM)**, pages 2405–2410, 2014.