BlindDB: an Encrypted, Distributed, and Searchable Key-value Store

Xingliang Yuan†, Xinyu Wang‡, Jianxiong Lin†, Cong Wang†, and Chen Qian†

†City University of Hong Kong, China; ‡University of Kentucky, United States

1. Introduction

Modern distributed data stores also known as NoSQL databases are offering superior performance, incremental scalability, and fine availability for data-intensive computing and cloud-based applications. However, at the rise of big data, privacy concerns in the multi-tenant cloud environment become even more pressing than before. Recent works [2, 3] provided solutions on secure query over encrypted data, with various trade-offs among security, efficiency, functionalities, and data dynamic support. But how to build a full-fledged encrypted key-value store remains less clear. In this project, we focus on designing and implementing a scalable, private, and searchable key-value store.

2. Trending Features of Modern Key-value Stores

(a): One single back end supports multi-data models.

(b): Allows for data access also from other (secondary) attributes of data.

3. System Architecture

(a): System architecture.

(b): Envisioned framework.

4. Encrypted Key-value Store

Figure 1: NoSQL distributed systems

Figure 2: Rich features of modern key-value stores

Figure 3: Our proposed system architecture. We propose the encrypted and distributed key-value store, where the primary key is transformed by the pseudo-random function (PRF) and the value is encrypted via symmetric encryption. Here, each pair can still be distributed to different nodes evenly by the known consistent hashing algorithm.

Figure 4: The construction of encrypted key-value store. Building on top of the encrypted key-value store, our system can flexibly support multiple data models. As our first effort, the column-oriented data model is explored, which is widely adopted in NoSQL systems, e.g., Cassandra and HBase.

5. Encrypted Distributed Index

As prior encrypted indexes are not designed for distributed systems, they do not specifically consider the data and index locality. Even if it can be applied, painful communication overhead will be introduced since the data and the index are accessed on different nodes. Besides, most of them cannot be incrementally scaled. When the number of data values exceeds the index capacity, the rebuilding of the entire index will be inevitable.

Figure 5: Encrypted distributed local index. For the column-oriented data model, we model the encrypted indexes the data records with the same attributes. Here, we adopt the encrypted index construction in [3] for its space efficiency and easy implementation, but integrate our proposed secure feature partition algorithm to design a group of fully distributed local indexes.

6. Performance Evaluation

We implement the system prototype and deploy it to Microsoft Azure. We create a Redis cluster that consists of 4 Standard P12 instances as the nodes of the encrypted KV store and 9 Standard A4 instances as the clients of data applications. Each Standard P12 instance is assigned with 4 vCores, 28GB RAM and 200GB SSD, and each Standard A4 instance is assigned with 8 vCores, 140GB RAM, and 200GB SSD. Each instance runs with up to 10 threads to generate the workload for performance and scalability evaluation.

Figure 6: Performance evaluation of the proposed system. The throughput of Put and Get have 27% and 28% loss respectively to non-encrypted Redis. Put and Get latency increase gradually when the workload reaches 80% of maximum throughput of non-encrypted Redis.

7. Concluding Remarks

We design an encrypted distributed key-value store that embraces a bunch of prominent features of plaintext NoSQL systems while ensuring data security. To support secure attribute queries, we then propose encrypted local indexes considering security, efficiency, and data locality. The results demonstrate that our encrypted data store can be accessed efficiently.

8. Acknowledgment

This work was supported in part by Research Grants Council of Hong Kong (Project No. CityU 138513), grant from City University of Hong Kong (Project No. 7004279), and a research grant award from Microsoft Azure.

References

