Autonomic, Elastic, Fault-tolerant, Scalable, and Secure Data Management in the Cloud

Sudipto Das, Aaron Elmore, Shiyuan Wang, Divyakant Agrawal, Amr El Abbadi

Abstract: Cloud computing has emerged as a revolutionary computing paradigm enabled by economies of scale due to large scale operations, pay-per-use pricing, and the commoditizing of computing resources. Database management systems (DBMSs) powering data-rich applications deployed in the cloud face a unique set of challenges calling for novel techniques, algorithms, and designs for cloud DBMSs. We present an overview of our research addressing different challenges encountered by the next generation of database management systems.

DBMSs powering cloud application platforms must serve large numbers of applications with unpredictable load patterns while minimizing the operating cost leveraging the underlying pay-per-use infrastructure. We have designed ElasTraS, an Elastic TraNsectional relational database for cloud platforms. ElasTraS is a confluence of two major design philosophies: traditional relational database systems (RDBMS) that allow the efficient execution of OLTP workloads and provide ACID guarantees for small databases and the Key-value stores that are elastic, scalable, and highly available. Effective resource sharing and the consolidation of multiple tenants on a single server allows ElasTraS to efficiently manage tenants with small data and resource requirements, while advanced database partitioning and live migration allows it to serve tenants that grow big, both in terms of data as well as load. ElasTraS achieves low overhead on-demand elasticity using live migration of tenant databases allowing expansion of the cluster size during high load and consolidation during usage troughs.

Live and on-demand migration of a tenant’s database with low performance impact and minimal service interruption is critical for efficient elastic scaling and load balancing. We designed live migration techniques for the two most common database architectures. On one hand, Iterative Copy—a technique for shared storage architectures—migrates the database cache and the state of active transactions ensuring minimal impact on transaction execution while allowing transactions active during migration to continue execution. Iterative Copy guarantees serializability for transactions active during migration while ensuring correctness during failures. On the other hand, Zephyr—a technique for shared nothing architectures —uses phases of on-demand pull and asynchronous push of data, requires minimal synchronization, and results in minimal service interruption with few or no aborted transactions, while minimizing the data transferred, and providing ACID guarantees during migration.

Another challenge is to provide consistent access to data for applications that frequently access groups of data items—collaboration driven applications such as online gaming, social networks, and collaborative editing are prime examples. We proposed the Key Group abstraction that defines a relationship between groups of keys and which represents the granule for on-demand transactional access. The Key Grouping protocol leverages the Key Group abstraction to collocate control for the keys in the group to allow efficient access to the group of keys. Using the Key Grouping protocol, we have designed and implemented G-Store that uses a key-value store as an underlying substrate to provide efficient, scalable, and transactional access to groups of keys.

To enable economies of scale, multiple applications need to share the same system resources. Guaranteeing individual privacy while using public information services and ensuring security of sensitive data in the cloud is critical. Our work addresses two aspects of data security: privacy of querying as well as privacy of data. Since users' queries may contain sensitive data as parameters for extracting information from public data sources, there is a potential risk in revealing or exposing users' intentions to malicious third parties or other competitors. Similarly, businesses want to store data in an obfuscated form in the cloud but still be able to query and manage their data in a secure and efficient way. To protect users’ query privacy, we leverage from Private Information Retrieval (PIR) techniques and propose using bounding boxes to limit the expensive computation often associated with PIR. To support common database queries and data processing tasks on encrypted relational data, we propose using a secure B+-tree index. We prevent potential inference attacks associating encrypted index with plaintext data by adding salt in the encoding process and routing requests via proxies. Moreover, we distribute data pieces using the Information Dispersal Algorithm (IDA), which makes it even more difficult to gather data for inference.