Enhancing geo-replicated cloud storage with database replication at edge servers

Geo-replication has emerged as an important technique for cloud services over the Internet. Geo-replication copies data to multiple data centers to improve performance, by avoiding slow long-haul communication, and to improve availability and fault tolerance, thanks to redundancy. However, user requests still incur at least one round-trip across Wide Area Network (WAN). While this might be acceptable for some applications such as web browsing, an important class of applications (e.g., e-commerce applications) is actually latency sensitive. Amazon reported that every 100ms slowdown degrades revenue by one percent. Wide-area network communication remains a fundamental barrier to deliver on the promise of cloud. It becomes more challenging as the need for always-available and real-time applications are increasing. Staying on this path hinders the actual potential of cloud computing, where WAN communication is not ignorable. Latency has conflict with other current network targets such as security and energy efficiency that makes it unlikely to achieve desired improvement. How can we enjoy the full benefits of cloud computing without being WAN-limited?

CDNs are used by major providers of internet services such as Facebook and Amazon to address this issue by caching and delivering content at the network edge. CDNs avoid WAN-based loss by bringing cloud applications closer to end-users. Rather than relying on a data center over WAN, user requests are tunneled into nearby edge servers; it is more operationally efficient to quickly meet changing demands. Today's CDNs are evolving from caching content web pages to run application logic, a service is called edge computing. However, they only support restricted classes of data such as read-only databases, blind writes or collectively writes, so that general-purpose shared read-write transactions are centralized at the data center. Li et al. propose a wide area replication model that enhances CDNs by moving a copy of database to the edge. They use 1-copy-snapshot isolation (1CSI) to keep these distributed copies of data consistent. Read-only transactions commit locally at the edge servers, while a verification phase is needed for update transactions to ensure conflicts do not occur when concurrent clients update the same data items. The verification phase adds WAN overhead degrading the benefits of using CDNs for interactive applications. Commutable updates are desirable because they do not need the verification phase. Conflict-free Replicated Data Type (CRDT) includes many useful data types, such as counters, sets, graphs, and sequences that avoid conflicts when users independently modify shared objects. CRDTs are asynchronous, but cannot provide strong consistency guarantees, such as ensuring that a shared counter never goes negative. Moreover, in the presence of failures, system designers must choose to maintain either performance (and availability) or consistency – both are not possible together. To make the right balance between consistency and performance (and availability) is not trivial; it is application-dependent. In this work, we propose a new hybrid cloud model, called Database Delivery Network (DDN), that extends the geo-replication model with supporting both synchronous and asynchronous transactions at the network edge. Thus, it improves user-perceived latency by minimizing the number of synchronizations over WAN, while taking into account consistency requirements. Our model classifies updates into commutative (Blue), partially-commutative (Purple) and non-commutative (Red), and distinguishes the (global) states where partially-commutative operations can safely run asynchronously. Commutative updates can always run asynchronously, whereas partially-commutative updates in unsafe states and non-commutative operations are synchronous. We use reservation techniques to ensure operation in such states. A reservation promises, to a cache that holds it, that the system is in a state that allows the cache server to perform updates asynchronously.