Amazon Aurora Multi-Master

Justin Levandoski | Amazon Web Services | jjl@amazon.com

Amazon Aurora is a relational OLTP database service offered as part of Amazon Web Services (AWS). Aurora is based on a novel architecture consisting of an "upper" part of a database engine that pushes redo recovery down to a separate purpose-built, scale-out storage service. This design provides a number of advantages for running a hosted relational database service: reduction in network traffic, very fast crash recovery, failover to replicas without without data loss, and fault-tolerant, self-healing storage. A previous HPTS talk covered the details of Aurora's architecture that supports a single writer and multiple read replicas.

This talk provides an overview of Aurora's multi-master (Aurora MM) design that supports multiple writers in the same cluster on a shared storage volume. We first cover Aurora MM's design relative to other popular multi-writer designs. Aurora MM avoids distributed lock management, which tends to involve heavyweight synchronization and leads negative scaling effects. It also avoids a purely partitioned architecture that (a) requires a heavyweight consensus protocol for cross-partition transactions and (b) may require load balancing scheme due to "hot" partitions.

Aurora MM is in open preview and is based on a decoupled architecture whereby independent writers log database updates to the shared storage layer. Aurora MM uses an optimistic cross-writer resolution approach where the storage layer is the arbiter of conflict for all updates performed in the system. This means, for instance, that a transaction log write may fail, and the database must be prepared to handle such conflict. As an example, Figure 1 depicts a scenario where an orange writer and gray writer perform non-conflicting updates to page 1 and 3, respectively. However, both writers update page 2 for which the gray master wins quorum acknowledgment. In this case, the transaction for the gray master succeeds, while the orange master's transaction fails.

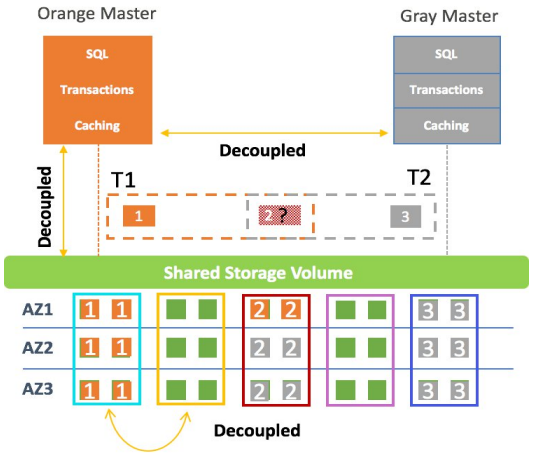

Figure 1. The Aurora Multi-Master Architecture

This approach has the advantage of requiring very little synchronization in the case of non-conflicting writes. It thus performs very well for cases where workloads naturally partition with little overlap. In addition to write scale, the approach also brings much higher availability to Aurora clusters: writes from a failed database can instantly be directed to other writers in the system. This talk also covers several features of Aurora MM such as global reads (providing consistent snapshot reads across the cluster) along with several performance optimizations made possible by the Aurora MM design.