Amazon Aurora

Design Considerations for High Throughput Cloud-Native Relational Databases

Ahmet Uysal

Tuesday 12th May, 2020

KOÇ UNIVERSITY

Agenda

1. Problems

2. Aurora[1] Architecture

3. Results

4. My Project Proposal

5. References

Bottlenecks of OLTP Databases

- Traditionally, read and write operations at individual devices constitutes the bottleneck.
  In cloud environment, I/O operations are spread to many nodes and disks.
  $\rightarrow$ The I/O bottleneck is moved to the network between Storage and Database layer
- Multi-phase synchronization protocols such as 2-phase commit (2PC) are intolerant of failures, and cloud systems continuously have soft and hard failures "**background noise**".

Key Ideas Behind Aurora Architechture

- **Layered Architecture**: Database and Storage are distinct layers, storage is implemented as an independent fault-tolerant service.
- Responsibilities are delegated to storage layer when possible. Redo logging, durable storage, crash recovery, and backup mechanisms are implemented in the storage layer.
- Minimize synchronous stalls and unnecessary writes/communication.

Network I/O Comparison

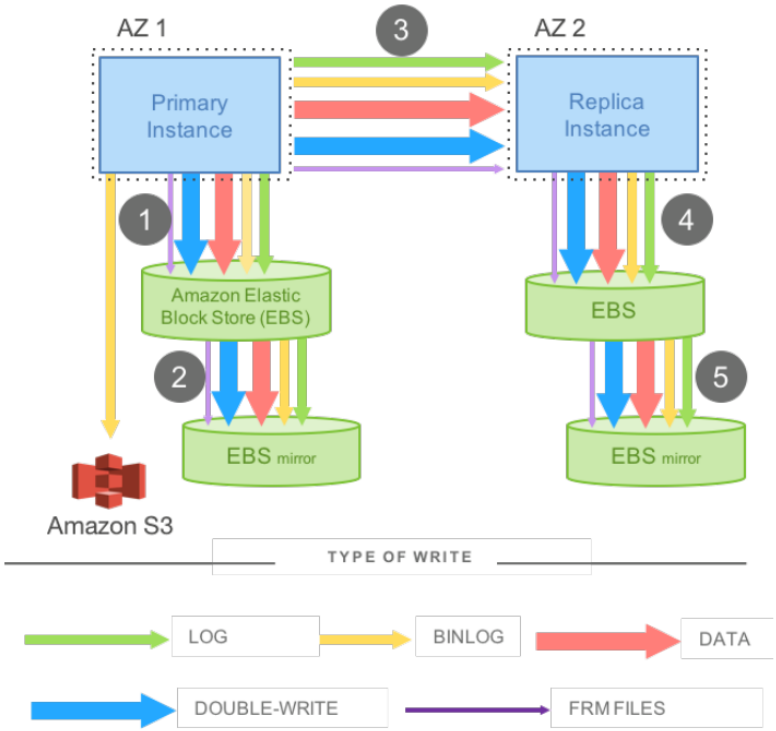

Figure: Network IO in mirrored MySQL

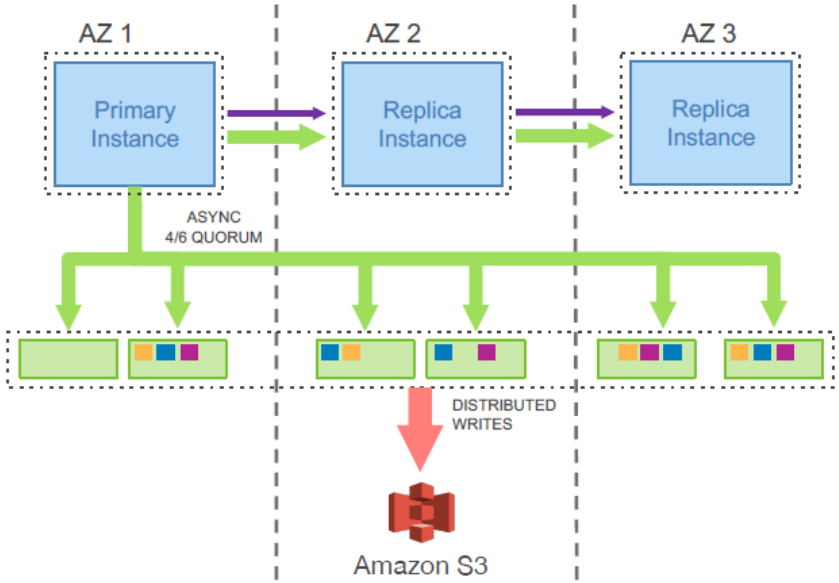

Figure: Network IO in Amazon Aurora

Durability

AWS Structure: AWS is divided into regions and regions are divided into **Availability Zones (AZ)**.
Aurora is designed to handle failure of an entire AZ and one additional node (AZ+1 fault-tolerance) without data loss, and has ability to write in the case of an AZ failure.
A quorum model with $V = 6$, $V_w = 4$ and $V_r = 3$ is used. This ensures every read have at least one voter from the most recent write quorum ($V_w + V_r > V$), and every write quorum have at least one member from the latest write quorum ($V_w > V/2$).

Durability

In order to further decrease the likelihood of failures, the design should ensure the probability of double fault is sufficently low over the time it takes to repair a failure (Mean time to Repair). MTTR is decreased by dividing database volume into small fixed size (10 GB) segments.

Role of Redo Log

In traditional databases, a **redo log** is generated when a data page is modified. These logs are used to construct after-image from before-image of the page.
In Aurora, “redo log is the database”. Redo logs are the only writes that cross the network between database and storage layers.
Storage layer **asynchronously** processes redo logs to create data pages.

Storage Layer Design

The main goal of Aurora's storage service design is to **minimize foreground write request latency**.
Majority of storage processing is run in background (asynchronously).
Unlike traditional systems, background processing has negative correlation with foreground processing.

Storage Node Activities

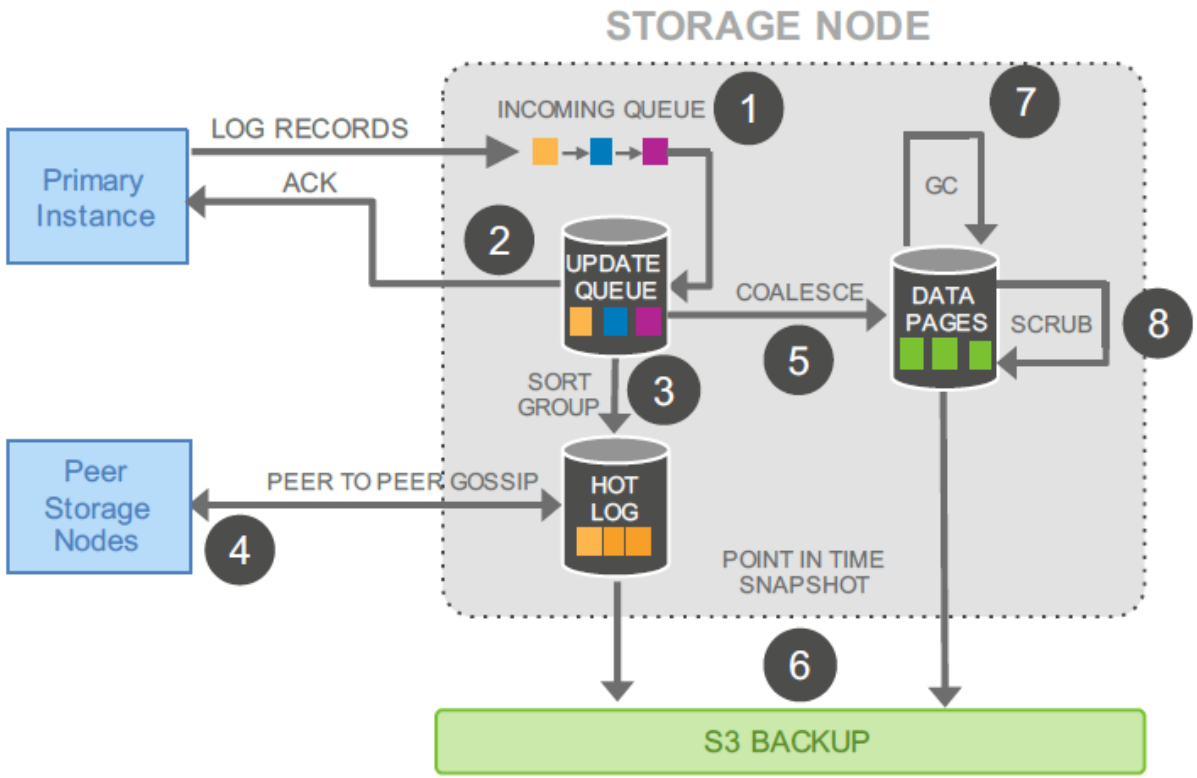

Figure: IO Traffic in Aurora Storage Nodes

Log Sequence Numbers (LSNs)

In Aurora consensus protocol is based on log sequence numbers. Points of consistency and durability are maintained in the database layer. These points are updated upon receiving acknowledge message from the storage.

Bird's Eye View

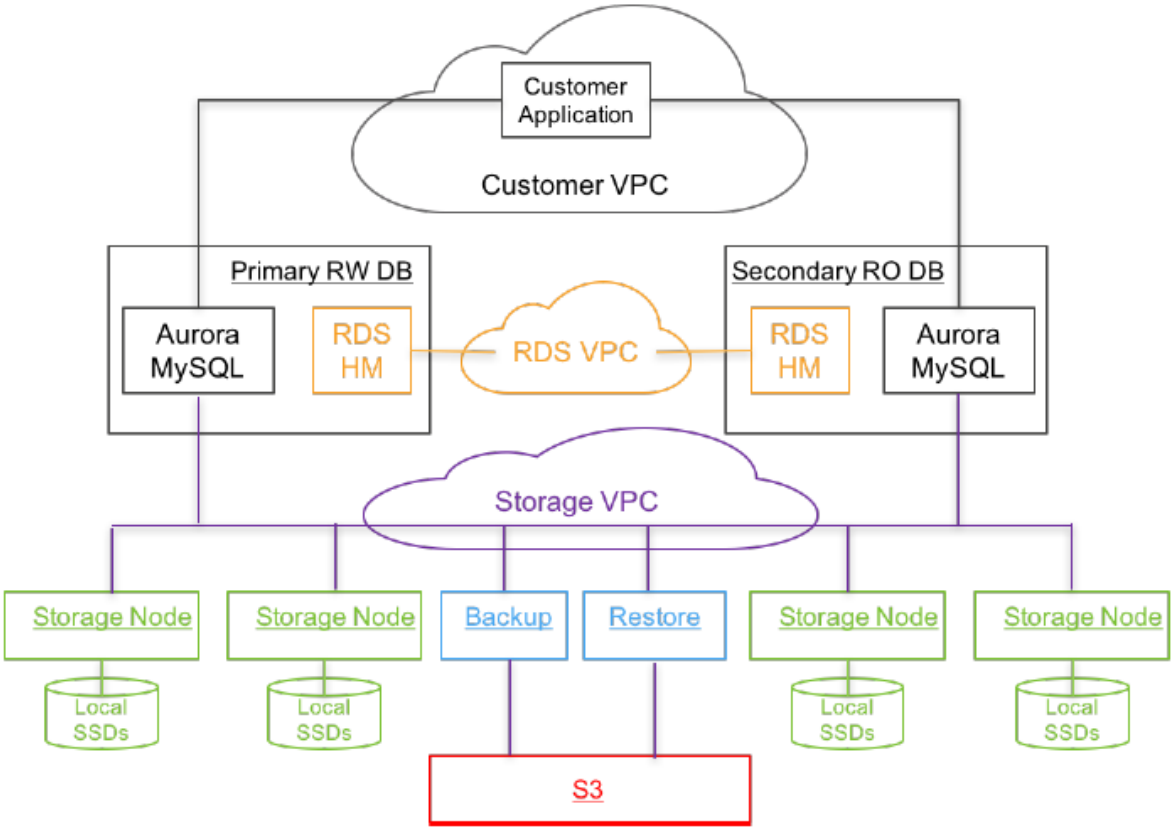

Figure: Aurora Architecture: A Bird's Eye View

Scaling With Instance Size (CPU and Memory)

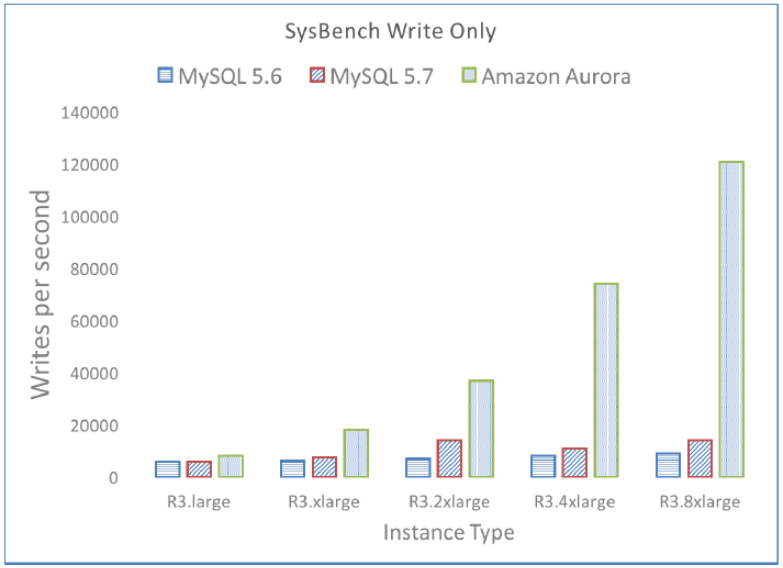

Figure: Comparison of write performance between Aurora and MySQL

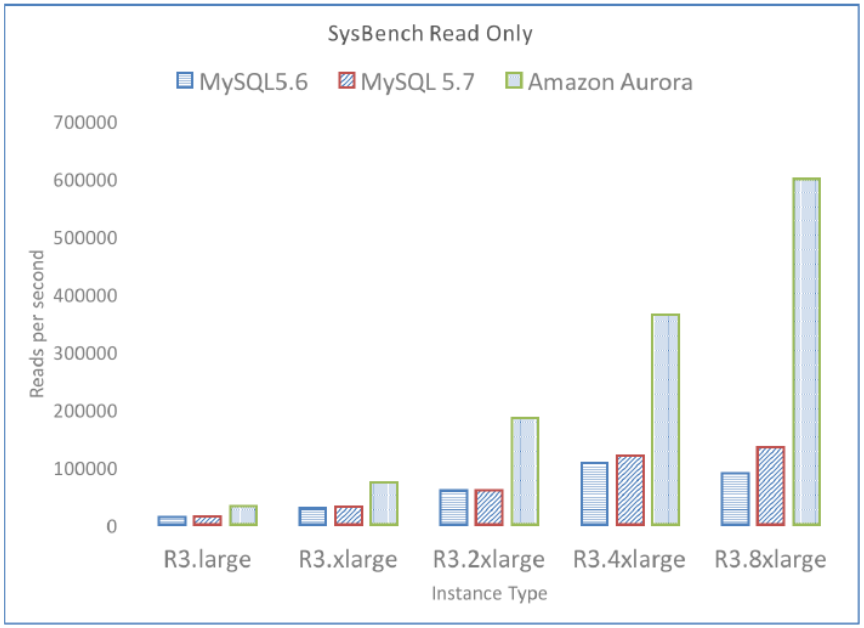

Figure: Comparison of read performance between Aurora and MySQL

Scaling with User Connections

Connections	Amazon Aurora	MySQL
50	40000	10000
500	71000	21000
5000	110000	13000

My Project Proposal

Apache Spark MLlib: Scalable machine learning library with many builtin algorithms.

References

[1] A. Verbitski, X. Bao, A. Gupta, D. Saha, M. Brahmadesam, K. Gupta, R. Mittal, S. Krishnamurthy, S. Maurice, and T. Kharatishvili, “Amazon aurora: Design considerations for high throughput cloud-native relational databases,” 05 2017, pp. 1041–1052.