Deep Dive on Amazon Aurora

Kevin Jernigan
kmj@Amazon.com
Principal Product Manager, Amazon Aurora
Amazon Web Services
<table>
<thead>
<tr>
<th>Time</th>
<th>Tutorials</th>
<th>Day 1</th>
<th>Day 2</th>
<th>List View</th>
<th>Grid View</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>MariaDB</td>
<td>MySQL</td>
<td>MongoDB</td>
<td>PostgreSQL</td>
<td>Cloud-Sponsored by AWS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Database Security &amp; Compliance</td>
</tr>
<tr>
<td>8:15 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Monitoring &amp; Ops</td>
</tr>
<tr>
<td>8:30 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Other GSDB</td>
</tr>
<tr>
<td>9:00 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:15 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:30 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:45 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:15 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:45 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:15 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:30 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:45 AM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:15 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:00 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:00 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:00 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:00 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:00 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Day 1**

- **8:00 AM** - Welcome Keynote
- **9:00 AM** - State of the Dolphins

**Day 2**

- **8:00 AM** - Amazon Relational Database Services (RDS)
- **9:00 AM** - TIDB 2.1, MySQL Compatibility, and Multi-Cloud Deployment
- **10:00 AM** - MyRocks in the Real World

**Break - Exhibit Hall Open**

- **10:30 AM** - What's new in and around MariaDB Server 10.3
- **11:00 AM** - MariaDB 10.1, Reverse Privileges (RCP)
- **11:30 AM** - MariaDB 10.3 vs MySQL 5.0
- **12:00 PM** - MariaDB 10.3 Optimizer and beyond

**Lunch - Exhibit Hall Open**

- **1:00 PM** - MariaDB Server 10.3 replication features
- **1:30 PM** - MariaDB 10.3 vs MySQL 5.0, replication features
- **2:00 PM** - MariaDB 10.3 vs MySQL 5.0, replication features
- **2:30 PM** - MariaDB 10.3 vs MySQL 5.0, replication features

**Coffee Break - Exhibit Hall Open**

- **3:00 PM** - MariaDB vs Oracle
- **3:30 PM** - MariaDB vs Oracle
- **4:00 PM** - MariaDB vs Oracle
- **4:30 PM** - MariaDB vs Oracle

**Follow-up**

- **5:00 PM** - MariaDB vs Oracle
- **5:30 PM** - MariaDB vs Oracle
- **6:00 PM** - MariaDB vs Oracle
- **6:30 PM** - MariaDB vs Oracle
Agenda

- Aurora overview
- Performance improvements
- Availability improvements
- Recent innovations
“Intuit invests significantly to own and operate high-end commercial databases underpinning our business. Until now, there just wasn’t a real alternative to obtain the reliability and performance our customers need. Amazon Aurora is a game-changer for us: providing the performance and availability features that rival expensive on-premises databases and SANs at a significantly lower price point. The RDS management capabilities on top of Amazon Aurora will allow us to focus our resources and energy on what matters most – building great applications and delighting our customers.”

Troy Otillio, Director, Public Cloud, Intuit
What is Amazon Aurora?
Database reimagined for the cloud

- **Speed** and **availability** of high-end commercial databases
- **Simplicity** and **cost-effectiveness** of open source databases
- Drop-in **compatibility** with MySQL and PostgreSQL
- Simple **pay as you go** pricing

Delivered as a **managed** service
Re-imagining the relational database

1. Scale-out, distributed architecture

2. Service-oriented architecture leveraging AWS services

3. Automate administrative tasks – fully managed service
Scale-out, distributed architecture

- Purpose-built log-structured distributed storage system designed for databases

- Storage volume is striped across hundreds of storage nodes distributed over 3 different availability zones

- Six copies of data, two copies in each availability zone to protect against AZ+1 failures

- Plan to apply same principles to other layers of the stack
Leveraging AWS services

**Lambda**
Invoke Lambda functions from stored procedures/triggers

**S3**
Load data from/ Select into S3, store snapshots and backups in S3

**IAM**
Use IAM roles to manage database access control

**CloudWatch**
Upload systems metrics and database logs
Automate administrative tasks

Schema design
Query construction
Query optimization

You

Automatic fail-over
Backup & recovery
Isolation & security
Industry compliance
Push-button scaling
Automated patching
Advanced monitoring
Routine maintenance

AWS

Takes care of your time-consuming database management tasks, freeing you to focus on your applications and business.
Aurora customer adoption

Fastest growing service in AWS history

Aurora is used by ¾ of the top 100 AWS customers
Who is moving to Aurora and why?

Customers using open source engines
- Higher performance – up to 5x
- Better availability and durability
- Reduces cost – up to 60%
- Easy migration; no application change

Customers using Commercial engines
- One tenth of the cost; no licenses
- Integration with cloud ecosystem
- Comparable performance and availability
- Migration tooling and services
Amazon Aurora is fast...

5x faster than MySQL; 3x faster than PostgreSQL
Aurora MySQL performance

MySQL SysBench results; R4.16XL: 64cores / 488 GB RAM

Aurora read write throughput compared to MySQL 5.6 based on industry standard benchmarks.
Aurora PostgreSQL performance

While running pgbench at load, throughput is 3x more consistent than PostgreSQL
How did we achieve this?

**DO LESS WORK**

- Do fewer IOs
- Minimize network packets
- Cache prior results
- Offload the database engine

**BE MORE EFFICIENT**

- Process asynchronously
- Reduce latency path
- Use lock-free data structures
- Batch operations together

DATABASES ARE ALL ABOUT I/O

NETWORK-ATTACHED STORAGE IS ALL ABOUT PACKETS/SECOND

HIGH-THROUGHPUT PROCESSING IS ALL ABOUT CONTEXT SWITCHES
Aurora I/O profile

MySQL I/O profile for 30 min Sysbench run
- 780K transactions
- 7,388K I/Os per million txns (excludes mirroring, standby)
- Average 7.4 I/Os per transaction

Aurora IO profile for 30 min Sysbench run
- 27,378K transactions
- 0.95 I/Os per transaction (6X amplification)
- 7.7X LESS

AMAZON AURORA

Primary Instance

Replica Instance

AZ 1

AZ 2

AZ 3

EBS

Amazon Elastic Block Store (EBS)

Amazon S3

EBS mirror

MYSQL WITH REPlica

Primary Instance

Replica Instance

AZ 1

AZ 2

EBS

Amazon Elastic Block Store (EBS)

Amazon S3

EBS mirror

ASYNC 4/6 QUORUM

DISTRIBUTED WRITES

TYPE OF WRITE

LOG
BINLOG
DATA
DOUBLE-WRITE
FRM FILES
IO traffic in Aurora (storage node)

IO FLOW

1. Receive record and add to in-memory queue
2. Persist record and ACK
3. Organize records and identify gaps in log
4. Gossip with peers to fill in holes
5. Coalesce log records into new data block versions
6. Periodically stage log and new block versions to S3
7. Periodically garbage collect old versions
8. Periodically validate CRC codes on blocks

OBSERVATIONS

All steps are asynchronous
Only steps 1 and 2 are in foreground latency path
Input queue is 46X less than MySQL (unamplified, per node)
Favor latency-sensitive operations
Use disk space to buffer against spikes in activity

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Performance enhancements in Aurora

DML throughput
- Adaptive thread pool
- Smart thread scheduler
- Asynchronous group commit
- Latch-free lock manager
- Lock compression
- Latch-free read views
- Fast B-Tree inserts
- Z-order spatial indexes
- Smart read-node selector
- Logical read ahead
- NUMA aware scheduler
- Highly concurrent catalog

Query execution
- Hash joins
- Batched scans
- Asynchronous key prefetch

DDL and Ops
- Instant schema update
- Faster index build
- High-performance auditing
What about availability?

“Performance only matters if your database is up”
6-way replicated storage
Survives catastrophic failures

Six copies across three availability zones
4 out of 6 write quorum; 3 out of 6 read quorum
Peer-to-peer replication for repairs
Volume striped across hundreds of storage nodes
Up to 15 promotable read replicas across multiple availability zones

- Re-do log based replication leads to low replica lag – typically < 10ms
- Reader end-point with load balancing and auto-scaling
Availability enhancements in Aurora

Unplanned unavailability
- Instant crash recovery
- Survivable cache
- Fast failover, incl. driver support

Planned unavailability
- Zero-downtime patching

Business continuity planning
- Continuous automated backup
- Point-in-Time-Restore
- Backtrack
- Cross-region read replicas
**Instant crash recovery**

**TRADITIONAL DATABASE**

Have to replay logs since the last checkpoint

Typically 5 minutes between checkpoints

Single-threaded in MySQL; requires a large number of disk accesses

**AMAZON AURORA**

Underlying storage replays redo records on demand as part of a disk read

Parallel, distributed, asynchronous

No replay for startup

---

Crash at $T_0$ requires a re-application of the SQL in the redo log since last checkpoint

Crash at $T_0$ will result in redo logs being applied to each segment on demand, in parallel, asynchronously
Database fail-over time

0 - 5s – 30% of fail-overs

5 - 10s – 40% of fail-overs

10 - 20s – 25% of fail-overs

20 - 30s – 5% of fail-overs
Zero downtime patching

**Before ZDP**

User sessions terminate during patching

**With ZDP**

User sessions remain active through patching
Recent Innovations
# Fast database cloning

**Clone database without copying data**
- Creation of a clone is nearly instantaneous
- Data copy happens only on write – when original and cloned volume data differ

**Example use cases**
- Clone a production DB to run tests
- Reorganize a database
- Save a point in time snapshot for analysis without impacting production system.
Database backtrack

Backtrack brings the database to a point in time without requiring restore from backups

- Backtracking from an unintentional DML or DDL operation
- Backtrack is not destructive. You can backtrack multiple times to find the right point in time
How does backtrack work?

We keep periodic snapshot of each segment; we also preserve the redo logs.

For backtrack, we identify the appropriate segment snapshots.

Apply log streams to segment snapshots in parallel and asynchronously.
Online DDL: MySQL vs. Aurora

MySQL

▪ Full Table copy in the background
▪ Rebuilds all indexes – can take hours or days
▪ DDL operation impacts DML throughput
▪ Table lock applied to apply DML changes

Aurora

<table>
<thead>
<tr>
<th>table name</th>
<th>operation</th>
<th>column-name</th>
<th>time-stamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>add-col</td>
<td>column-abc</td>
<td>t1</td>
</tr>
<tr>
<td>Table 2</td>
<td>add-col</td>
<td>column-qpr</td>
<td>t2</td>
</tr>
<tr>
<td>Table 3</td>
<td>add-col</td>
<td>column-xyz</td>
<td>t3</td>
</tr>
</tbody>
</table>

▪ Use schema versioning to decode the block.
▪ Modify-on-write primitive to upgrade to latest schema
▪ Currently support add NULLable column at end of table
▪ **Add column anywhere and with default coming soon.**
## Online DDL performance

<table>
<thead>
<tr>
<th></th>
<th>Aurora</th>
<th>MySQL 5.6</th>
<th>MySQL 5.7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>On r3.large</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10GB table</td>
<td>0.27 sec</td>
<td>3,960 sec</td>
<td>1,600 sec</td>
</tr>
<tr>
<td>50GB table</td>
<td>0.25 sec</td>
<td>23,400 sec</td>
<td>5,040 sec</td>
</tr>
<tr>
<td>100GB table</td>
<td>0.26 sec</td>
<td>53,460 sec</td>
<td>9,720 sec</td>
</tr>
<tr>
<td><strong>On r3.8xlarge</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10GB table</td>
<td>0.06 sec</td>
<td>900 sec</td>
<td>1,080 sec</td>
</tr>
<tr>
<td>50GB table</td>
<td>0.08 sec</td>
<td>4,680 sec</td>
<td>5,040 sec</td>
</tr>
<tr>
<td>100GB table</td>
<td>0.15 sec</td>
<td>14,400 sec</td>
<td>9,720 sec</td>
</tr>
</tbody>
</table>
Perfomance Insights

Dashboard showing load on DB
- Easy
- Powerful

Identifies source of bottlenecks
- Top SQL

Adjustable time frame
- Hour, day, week, month
- Up to 35 days of data
Aurora Serverless
On-demand, auto-scaling database for applications with variable workloads

- Starts up on demand, shuts down when not in use
- Automatically scales with no instances to manage
- Pay per second for the database capacity you use
Parallel query processing

Aurora storage has thousands of CPUs
- Presents opportunity to push down and parallelize query processing using the storage fleet
- Moving processing close to data reduces network traffic and latency

However, there are significant challenges
- Data stored in storage node is not range partitioned – require full scans
- Data may be in-flight
- Read views may not allow viewing most recent data
- Not all functions can be pushed down to storage nodes
Parallel Query: Use cases

Orders of magnitude faster queries

Parallelism increases with data size

Reduced contention with OLTP workload

Analytic workloads on OLTP working set

• Analyze real-time data

• Avoid building ETL pipelines for ad-hoc queries

• A large number of concurrent analytic queries
“We noticed query time reduce from 32 minutes to 3 minutes.” – preview customer (online media company)
Query Optimizer produces PQ Plan and creates PQ context based on leaf page discovery

PQ request is sent to storage node along with PQ context

Storage node produces

- Partial results streams with processed stable rows
- Raw stream of unprocessed rows with pending undos

Head node aggregates these data streams to produce final results
Processing at storage node

Each storage node runs up to 16 PQ processes, each associated with a parallel query.

PQ process receives PQ context:
- List of pages to scan
- Read view and projections
- Expression evaluation byte code

PQ process makes two passes through the page list:
- **Pass 1:** Filter evaluation on InnoDB formatted raw data
- **Pass 2:** Expression evaluation on MySQL formatted data
What is Amazon Aurora?
Database reimagined for the cloud

- Speed and availability of high-end commercial databases
- Simplicity and cost-effectiveness of open source databases
- Drop-in compatibility with MySQL and PostgreSQL
- Simple pay as you go pricing

Delivered as a managed service
Rate My Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations
11:20 → 12:10
Matterhorn 2

Rate & Review

Feedback (optional)

Anonymously

SUBMIT