Instant and fine-grained scaling with Amazon Aurora Serverless v2

Anum Jang Sher (she/her)
Sr. Product Manager
AWS

Agenda

Why serverless?

Use cases

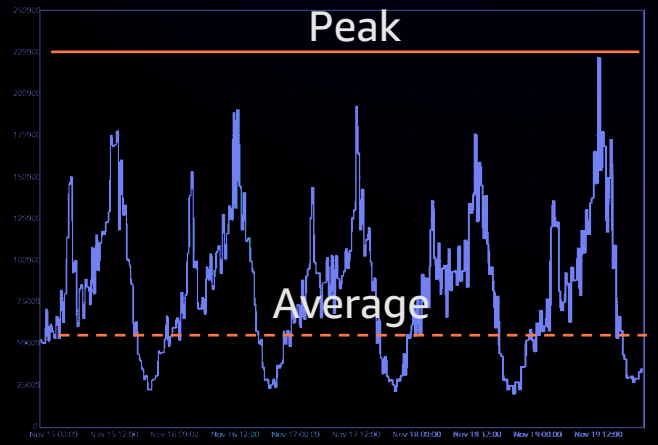
Aurora Serverless v2 deep dive

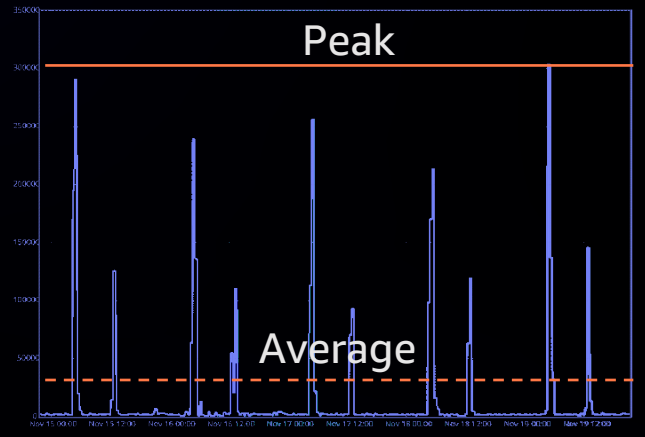
Demos

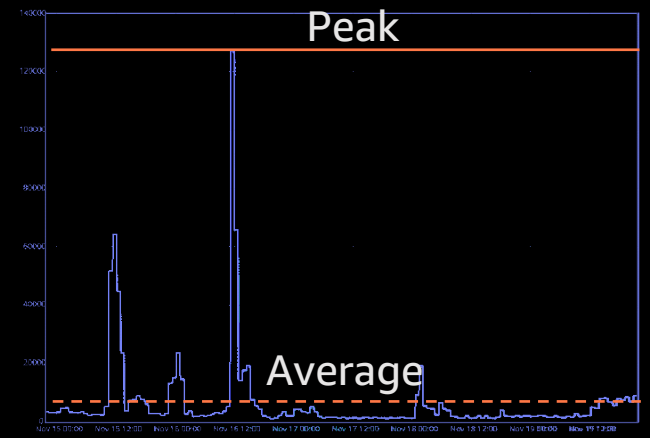
Getting started with Aurora Serverless v2

Database capacity: Cost vs. management

Variable and unpredictable workloads







Insufficient capacity → Experience degradation

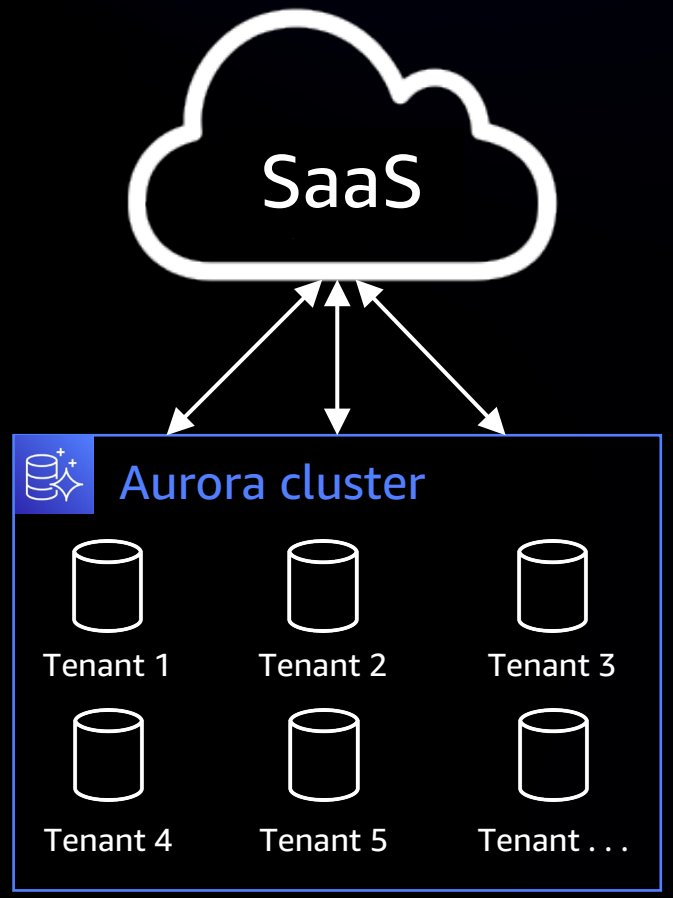
Provision for peak → Expensive

OR

Continuously monitor and scale → Difficult, requires experts, involves downtime

aws

Multi-tenant applications with per-tenant databases



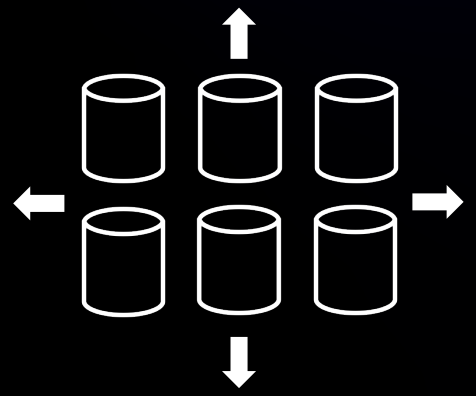
Thousands of end users, each with their own database

Colocate databases to improve utilization and cost efficiency

Tradeoff granularity of operational controls (e.g., backups and restores)

Requires continuous monitoring and shuffling due to busy workloads

Several application backed by databases



100s to 1,000s of databases

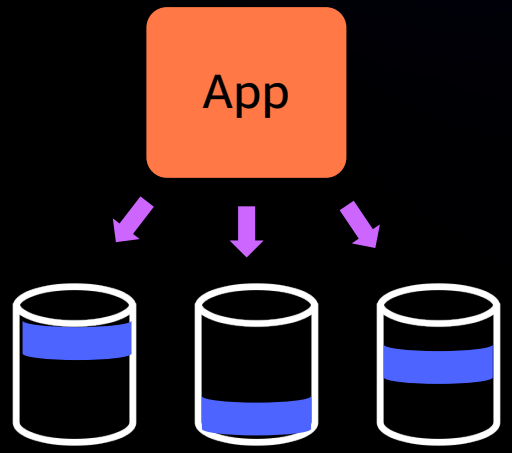
Customers with 100s or 1,000s of applications

Each application backed by one or more databases

Evolving application requirements require database capacity adjustments

Managing capacity for a database fleet on a budget is daunting

Scaled out databases



Applications needing write scalability split their database across multiple nodes for higher throughput

Predicting capacity for each node is hard and inefficient

Create too few nodes, you've to redistribute data taking downtime

Create too many nodes and pay for high costs as all nodes are not equally utilized

Amazon Aurora Serverless v2

On-demand and autoscaling configuration

Automatically scales capacity based on application needs

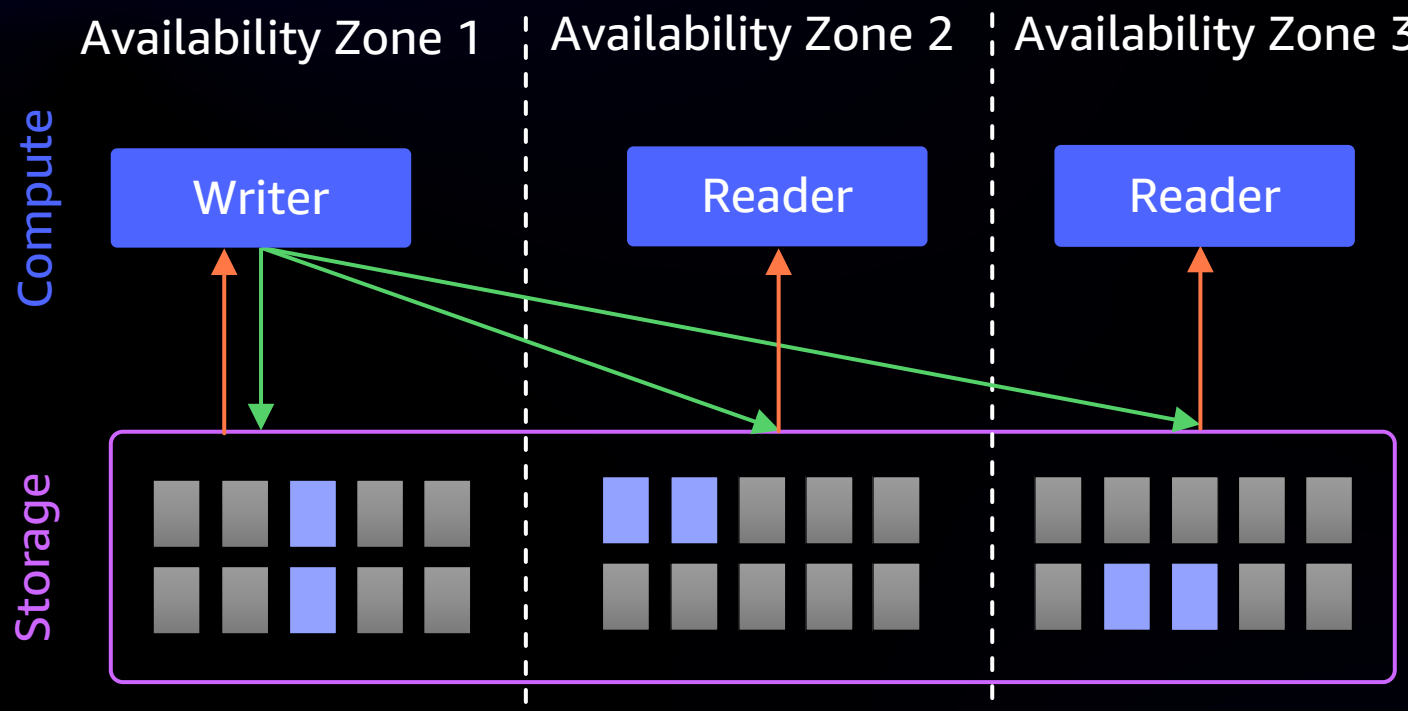
Simple **pay-per-use** pricing per second

Next version scales instantly to support demanding applications

Worry-free database capacity management

aws

Aurora architecture supports serverless



Purpose-built, log-structured, distributed storage designed for cloud databases

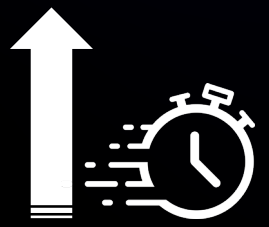
6 copies across 3 AZs for high availability, durability, and performance

Separation of storage and compute:

- Routine operations (e.g., backups) done without compute
- Up to 15 low-latency (<10 millisecond) read replicas
- **Compute scales independently**

Demo: Creating an Aurora Serverless v2 database

Demanding workloads require . . .



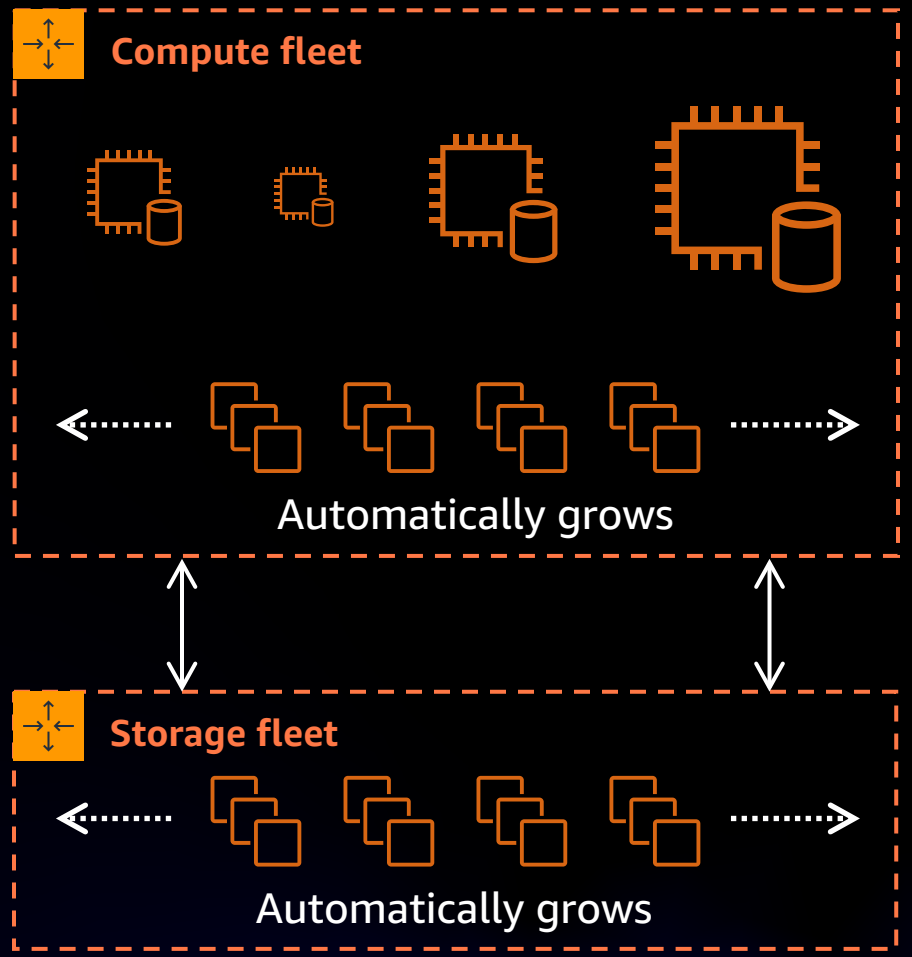
Instant and non-disruptive capacity scaling

Fine-grained and predictable capacity adjustments

High availability, disaster recovery, and enhanced capabilities

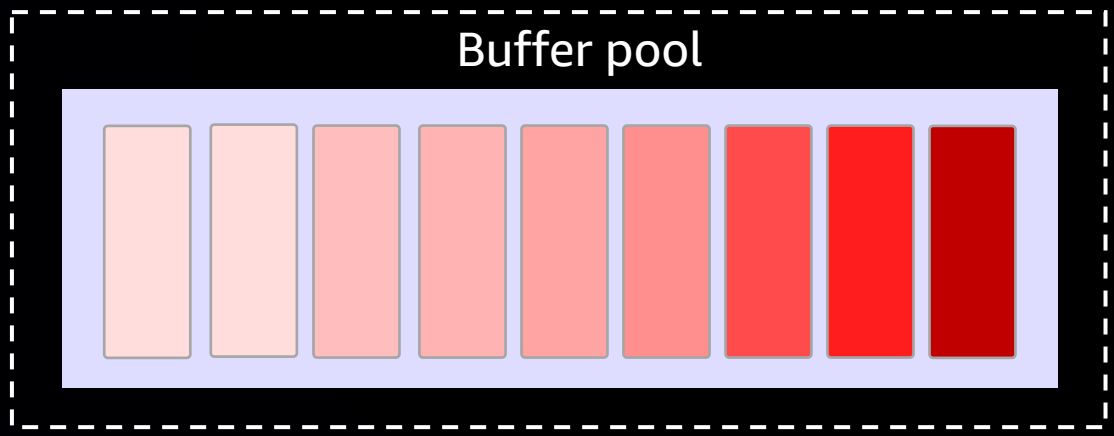
Instant, in-place scaling

- Scales in place in under a second by adding more CPU and memory resources
- No impact due to scaling even when running hundreds of thousands of transactions
- Compute fleet continuously monitored and scaled horizontally for heat management
- Background movement of idling instances while preserving state (e.g., buffer pool, connections)
- Up to 15x faster scale downs

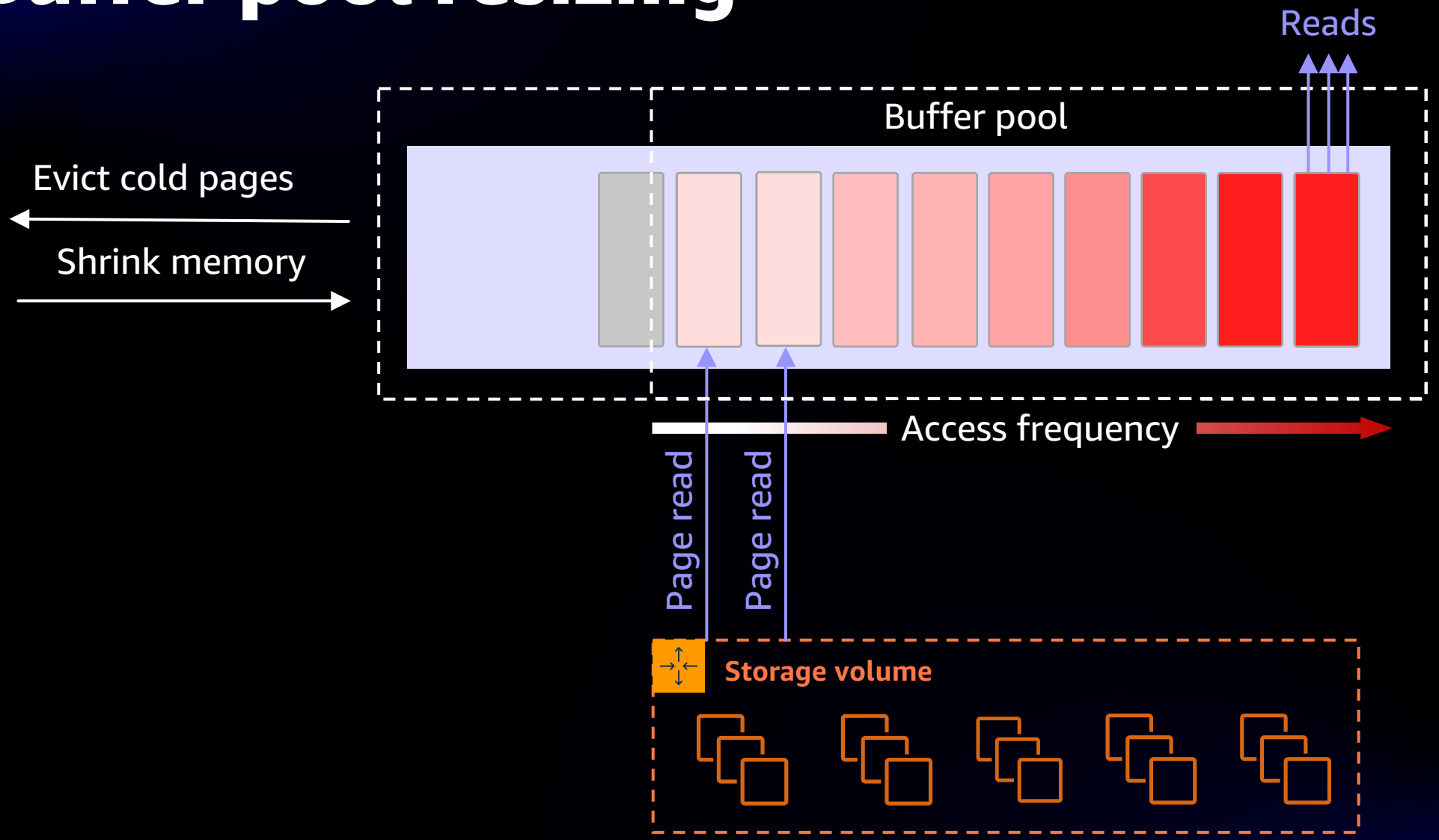


Buffer pool resizing

- Buffer pool size scaled along with database capacity
- Parameters automatically adjusted:
  - MySQL: innodb_buffer_pool_size
  - PostgreSQL: shared_buffers
- Memory allocation: 75% for buffer pool and 25% for heap
- Buffer pool scaled down through a combination of least frequently used (LFU) and least recently used (LRU) algorithms

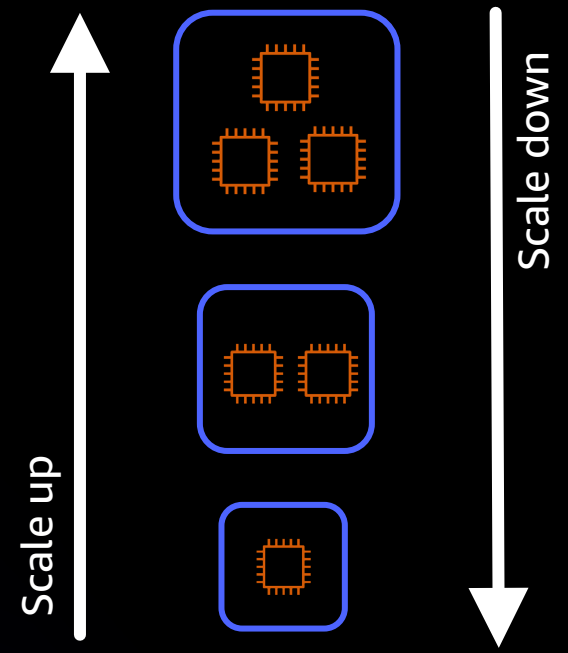


Buffer pool resizing



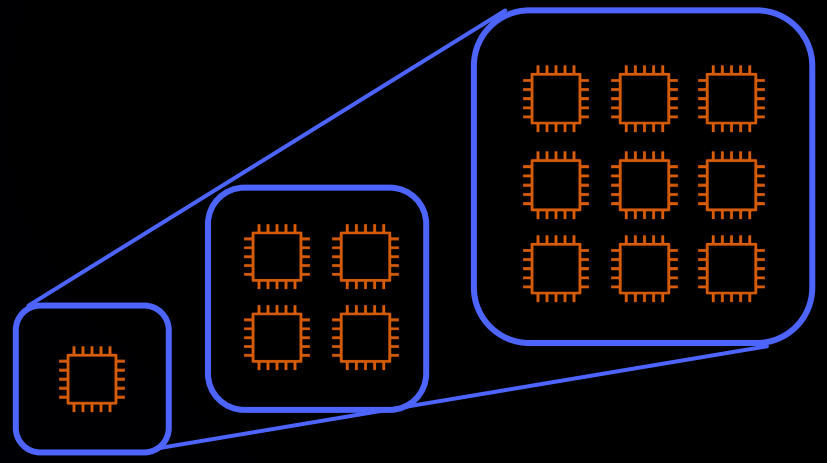
Fine-grained capacity adjustments

- Aurora Serverless capacity is measured in Aurora Capacity Unit (ACU)
- 1 ACU comes with 2 GiB of memory
- CPU and networking similar to what is available in provisioned Aurora instances
- Starting capacity as small as 0.5 ACU (1 GiB)
- Fine-grained scaling with as little as 0.5 ACU increments



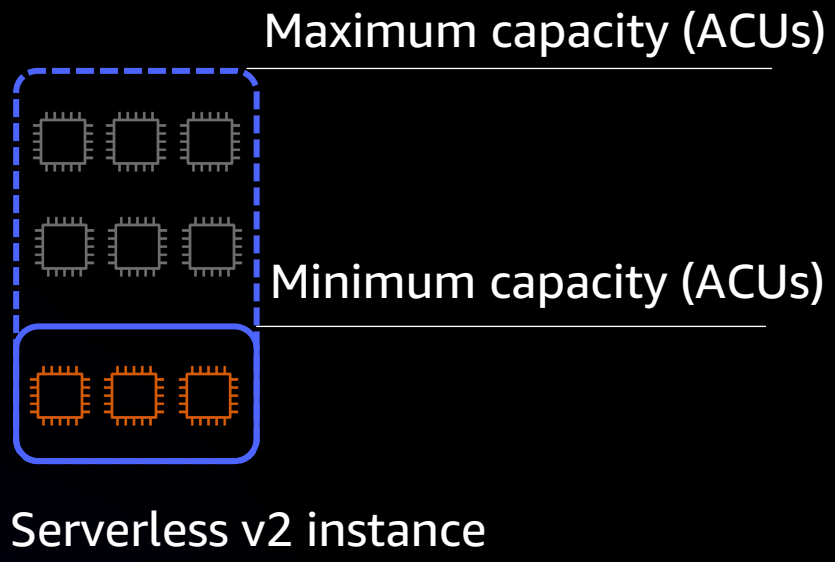
Scaling factors

- Scale-up rate is predictable and proportional to current capacity – larger instances scale up faster
- CPU utilization of both foreground and background processes (e.g., purge or vacuum)
- Memory utilization of internal data structures (e.g., buffer pool)
- Network throughput is proportional to capacity; capacity is scaled to match network throughput needs



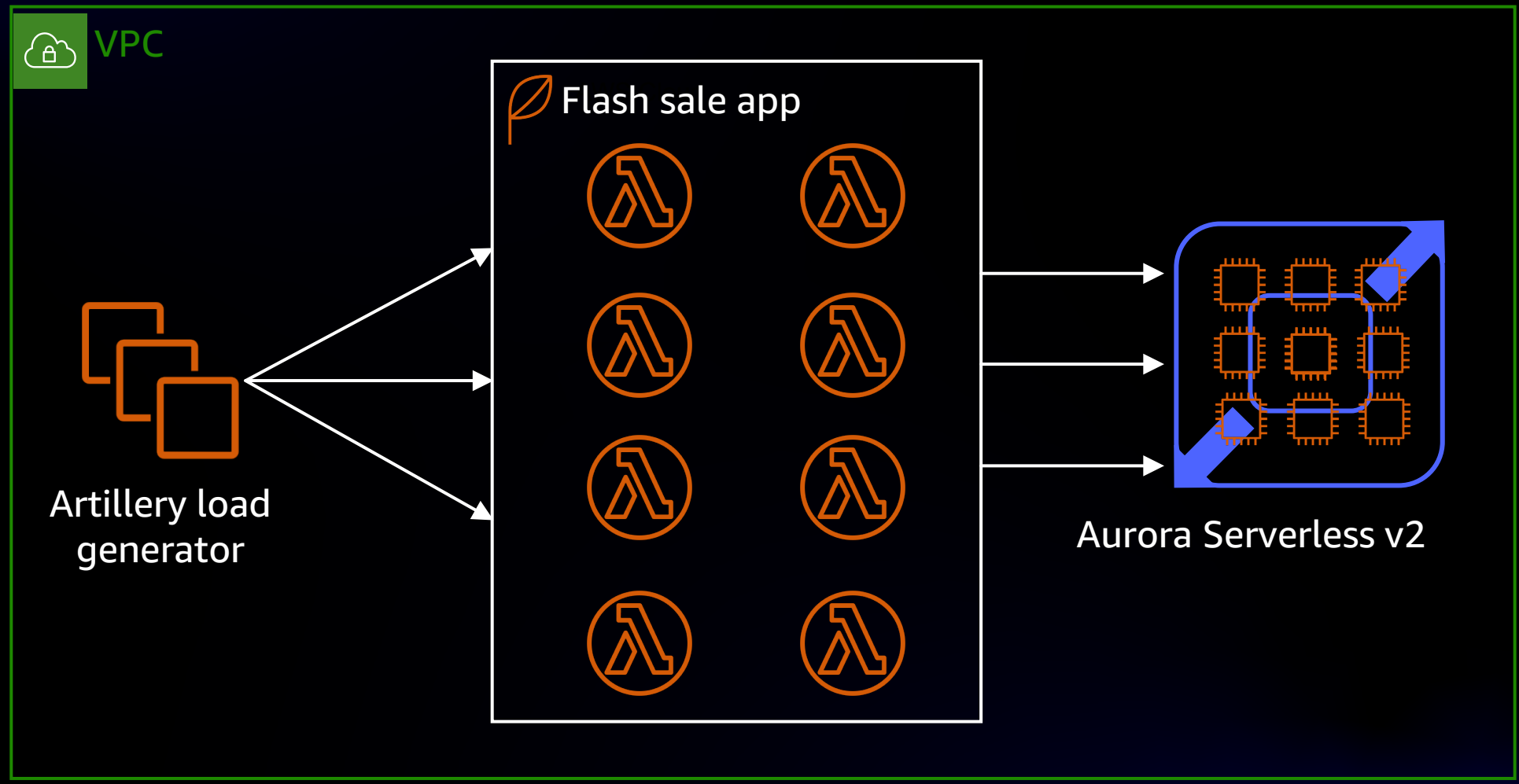
Configuring capacity

- Minimum capacity determines the starting capacity of the instance
- Adjust minimum capacity for a suitable scaling rate
- Increase minimum capacity ahead of anticipated spike
- Maximum capacity is a budget control
- Configure maximum to support peak workload and features (e.g., performance insights)



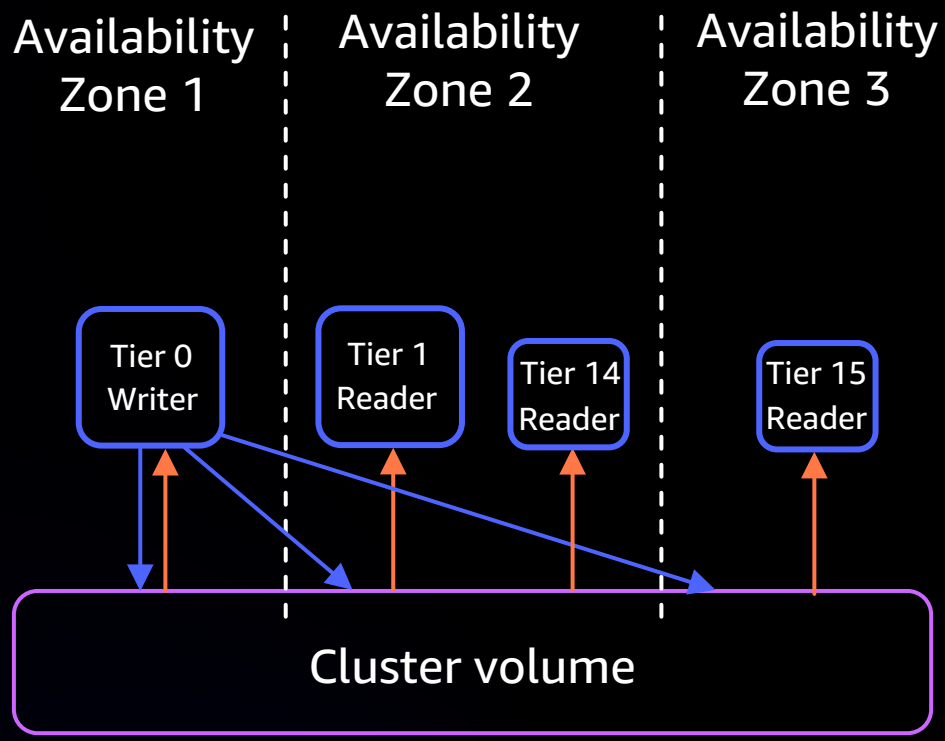
Demo: Scaling

Demo: Flash sale

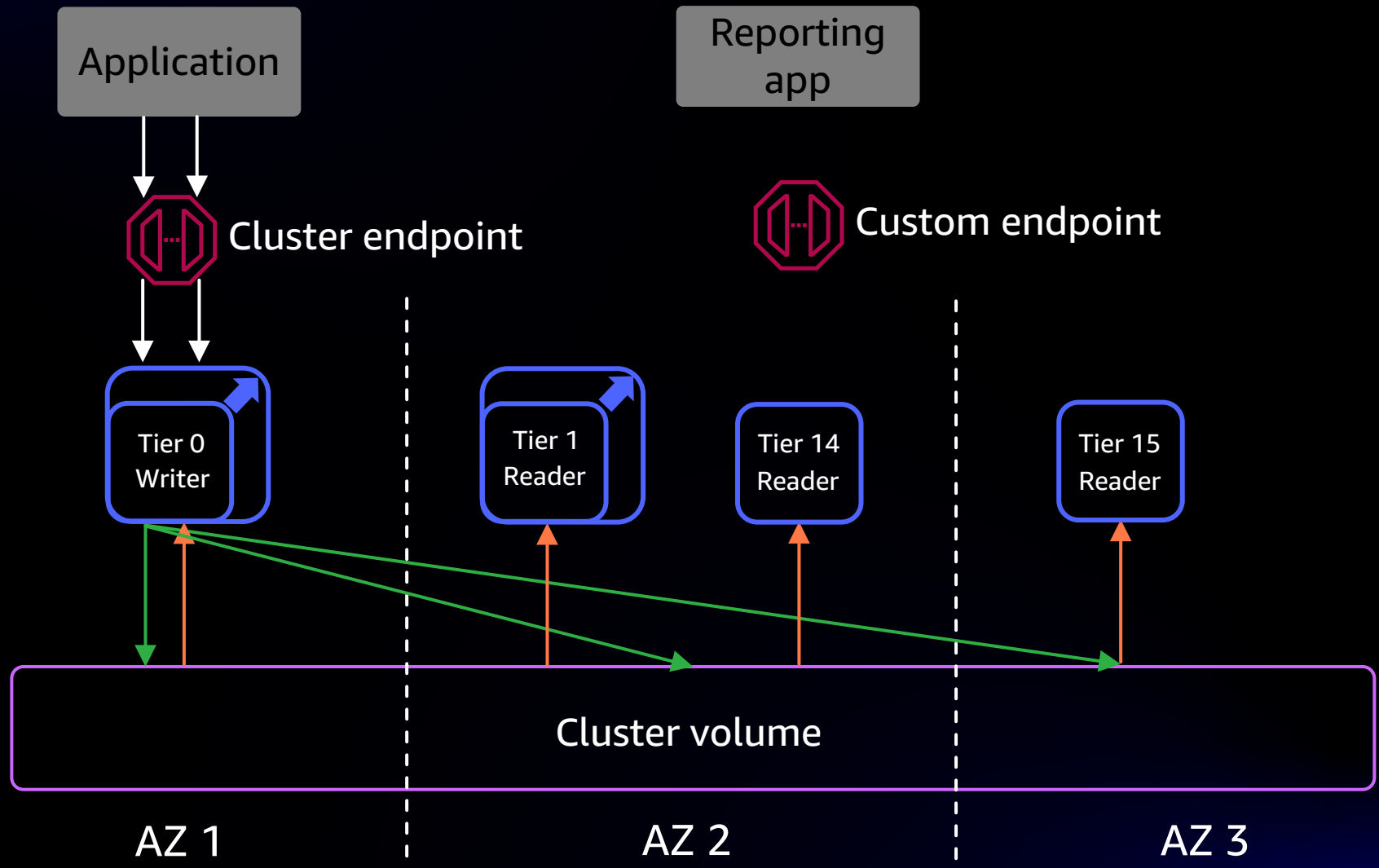


High availability and read scaling

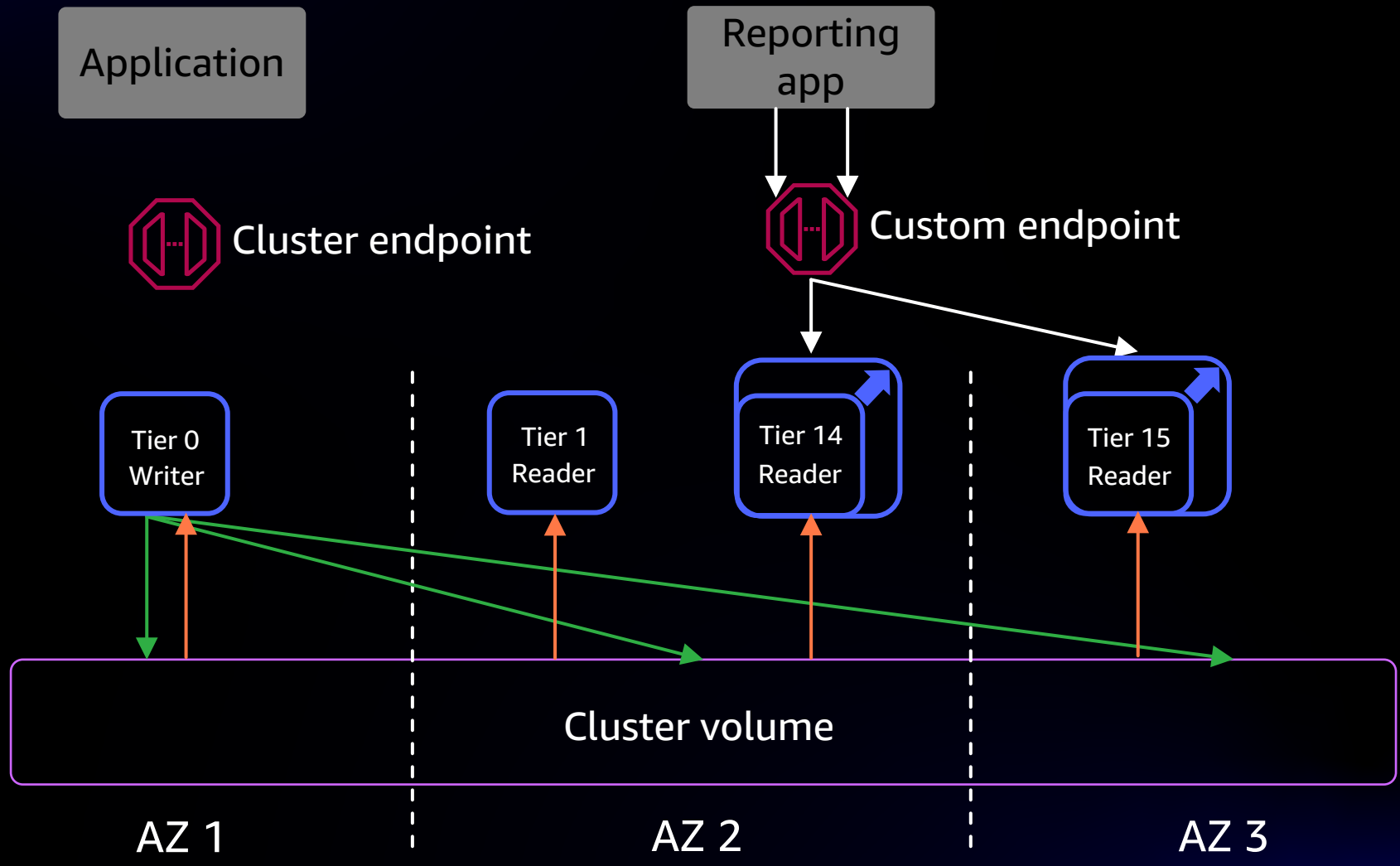
- Up to 15 read replicas act as failover targets
- All instances inherit capacity configuration from the cluster
- Tier 0 and 1 read replicas match the size of the primary instance
- Deploy across separate AZs
- Multi-AZ Aurora clusters supported by 99.99% uptime SLA



Multi-AZ high availability

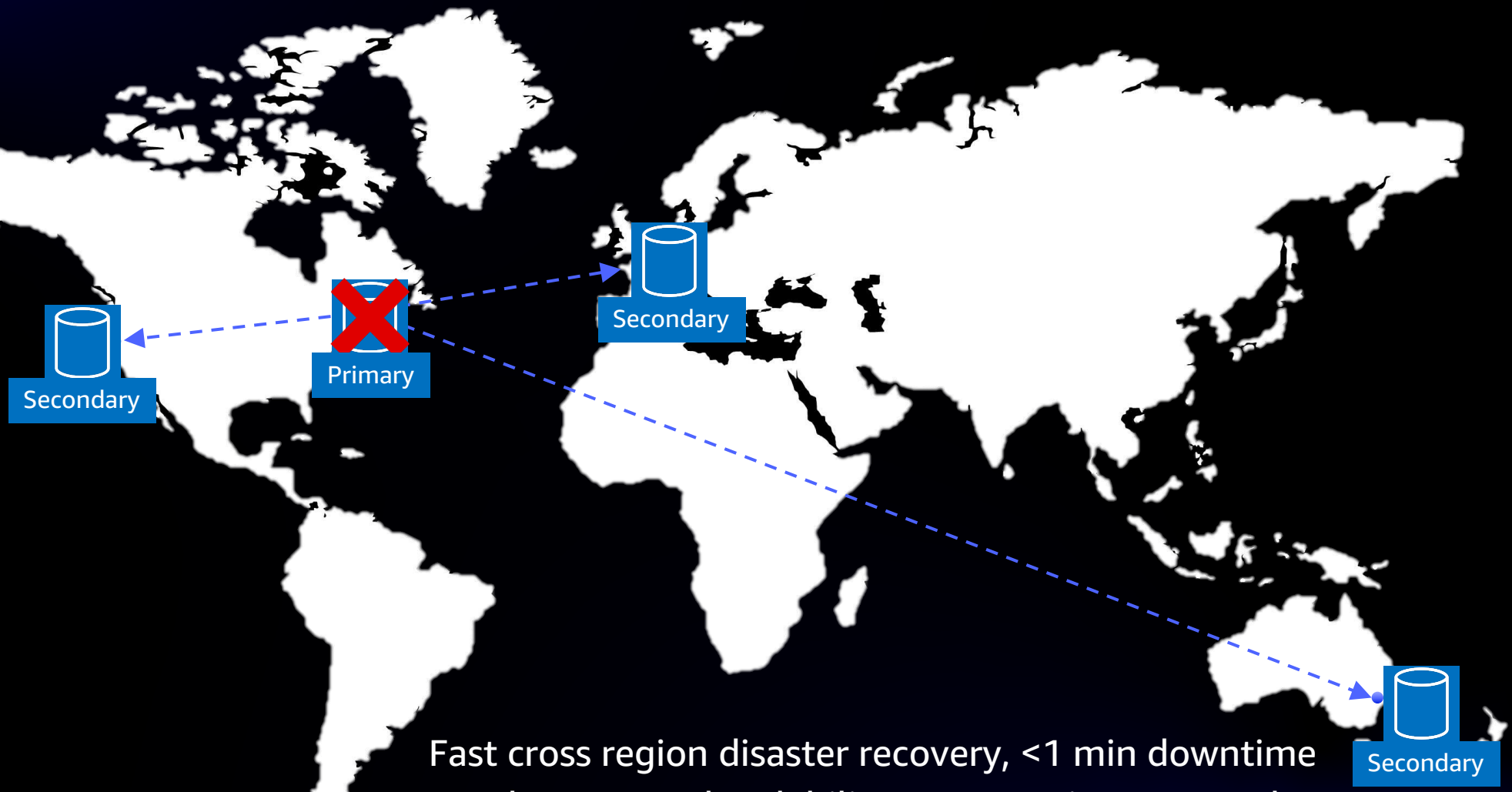


Read scalability



Demo: High availability with Aurora Serverless v2 read replicas

Aurora Global Database



Fast cross region disaster recovery, <1 min downtime

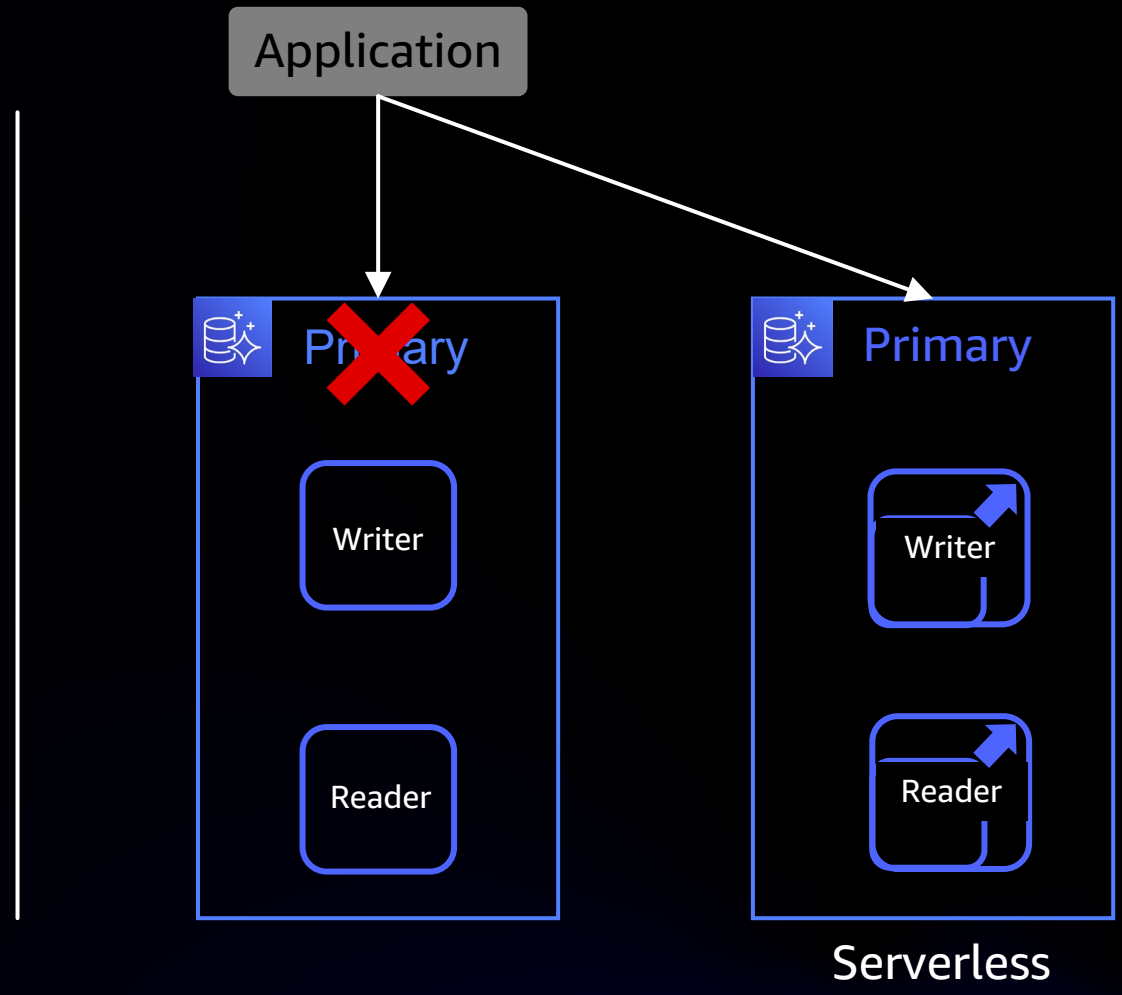
Low latency read scalability across regions, <1 sec lag

Up to 5 secondary regions, 90 read replicas

aws

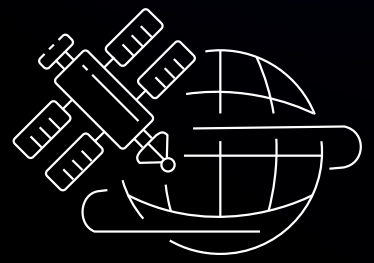
Serverless global database

- Configure secondary regions with Aurora Serverless v2
- Pay for only minimal capacity when idling
- Writer and readers scale up after failover
- Scale secondary regions independently for reads closer to end users



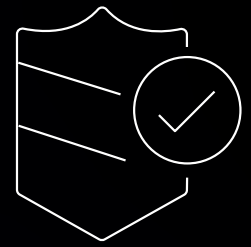
Amazon RDS Proxy

A FULLY MANAGED, HIGHLY AVAILABLE DATABASE PROXY FOR AMAZON RDS AND AMAZON AURORA

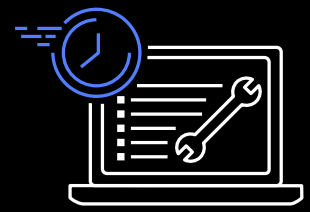


Pool and share DB connections for improved app scaling

Increase app availability and reduce DB failover times



Manage app data security with DB access controls



Fully managed DB proxy, compatible with your database

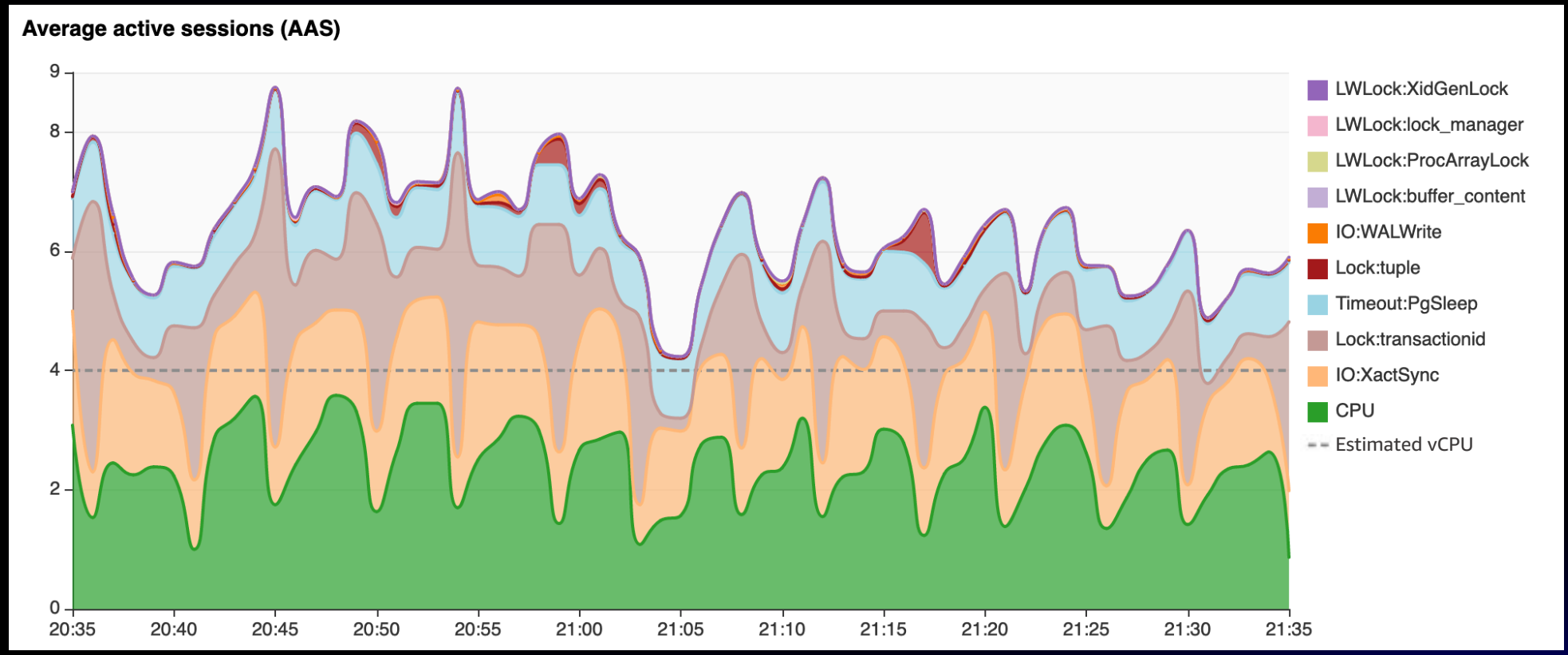
Amazon RDS Proxy supports Aurora Serverless v2, including mixed configurations with Aurora provisioned and serverless instances within a cluster

aws

Amazon RDS performance insights

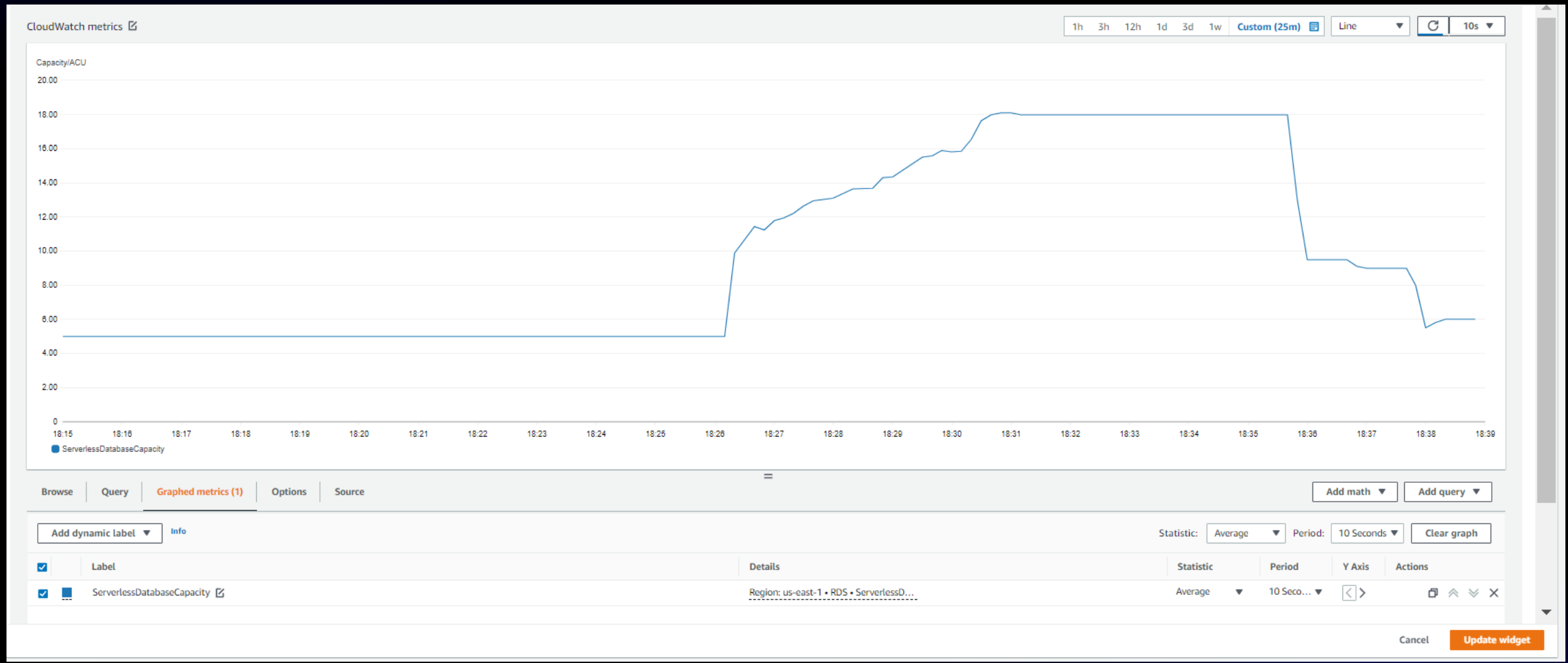
MONITOR AND TUNE DATABASE PERFORMANCE

- Easy and powerful dashboard to visualize database load
- Investigate database performance issues, even if you're not a database expert



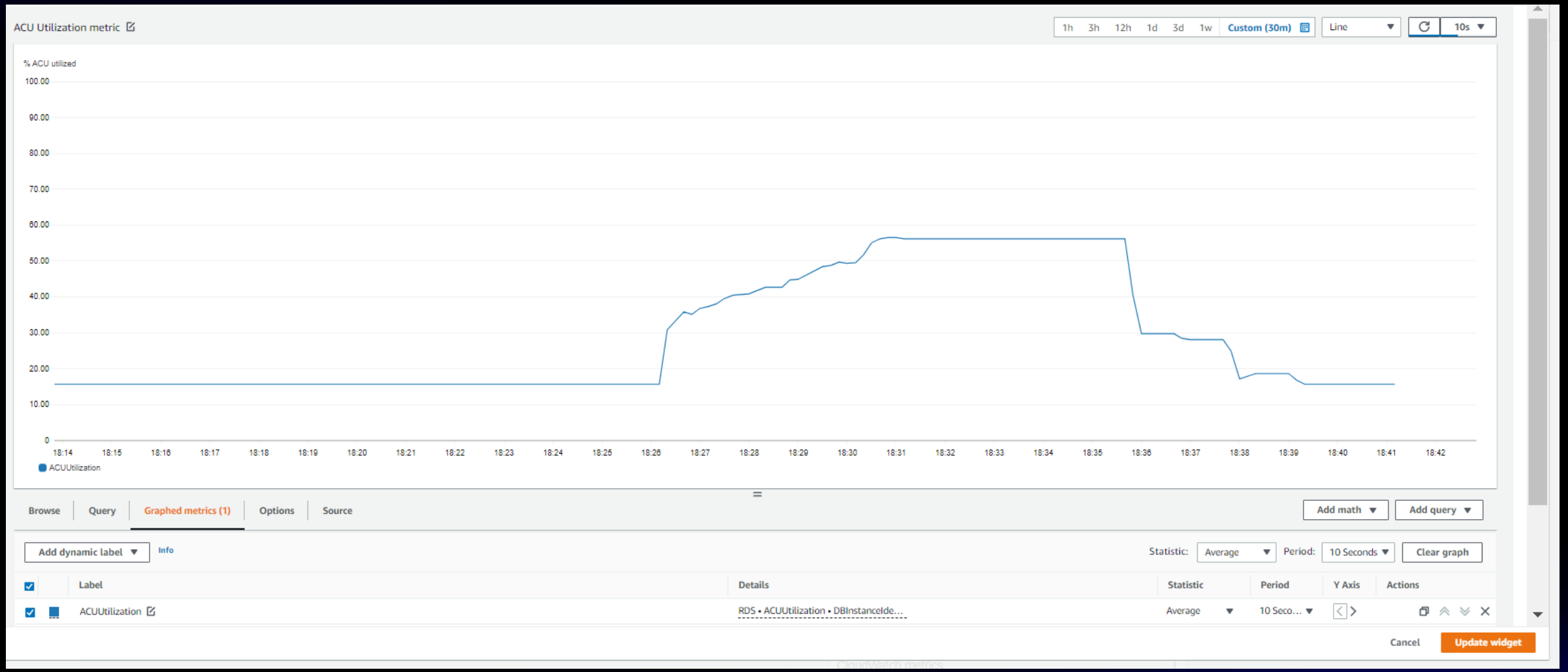
Amazon CloudWatch metrics

SERVERLESSDATABASECAPACITY (ACUS) – CURRENT CAPACITY OF THE INSTANCE IN ACUS



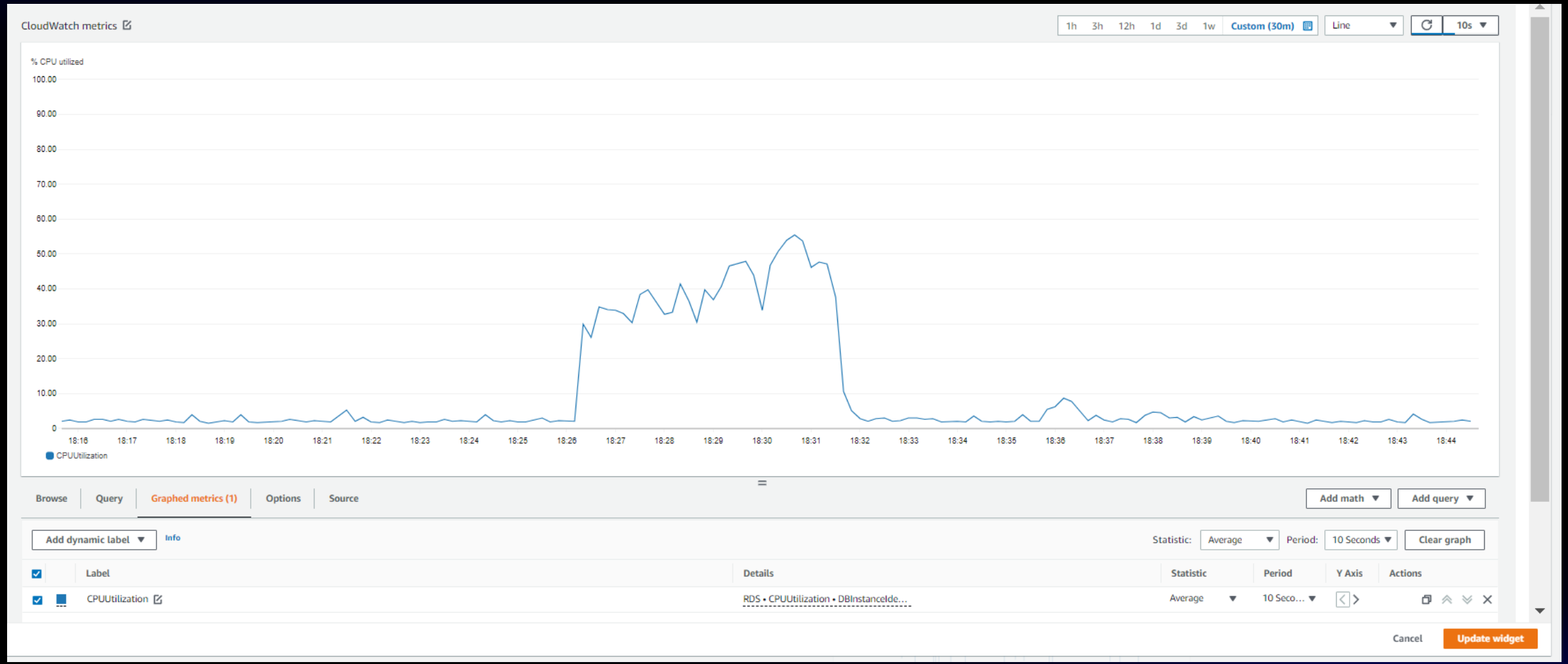
CloudWatch metrics

ACUUTILIZATION (%) – PERCENTAGE OF CURRENT CAPACITY OUT OF SPECIFIED MAXIMUM CAPACITY



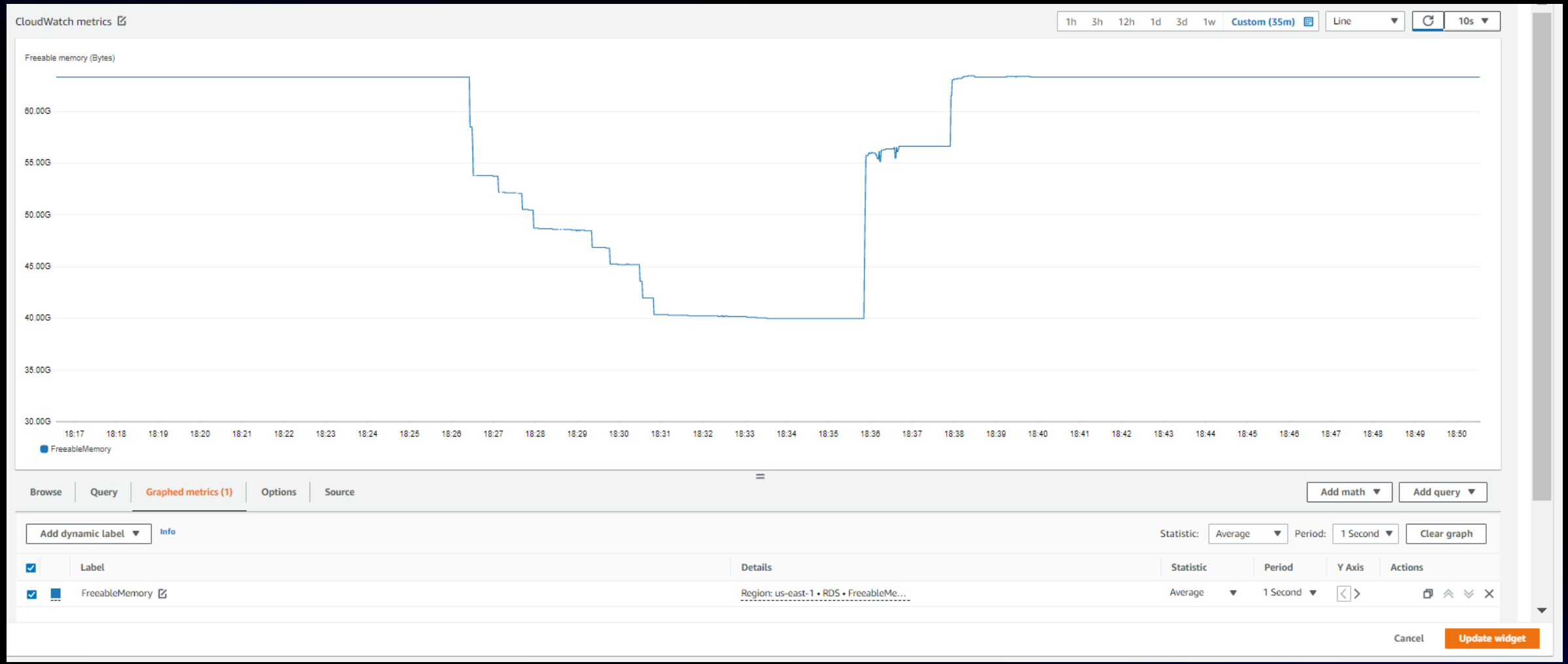
CloudWatch metrics

CPUUTILIZATION (%)– PERCENTAGE OF CPU, BASED ON MAXIMUM ACU, CURRENTLY USED BY THE INSTANCE



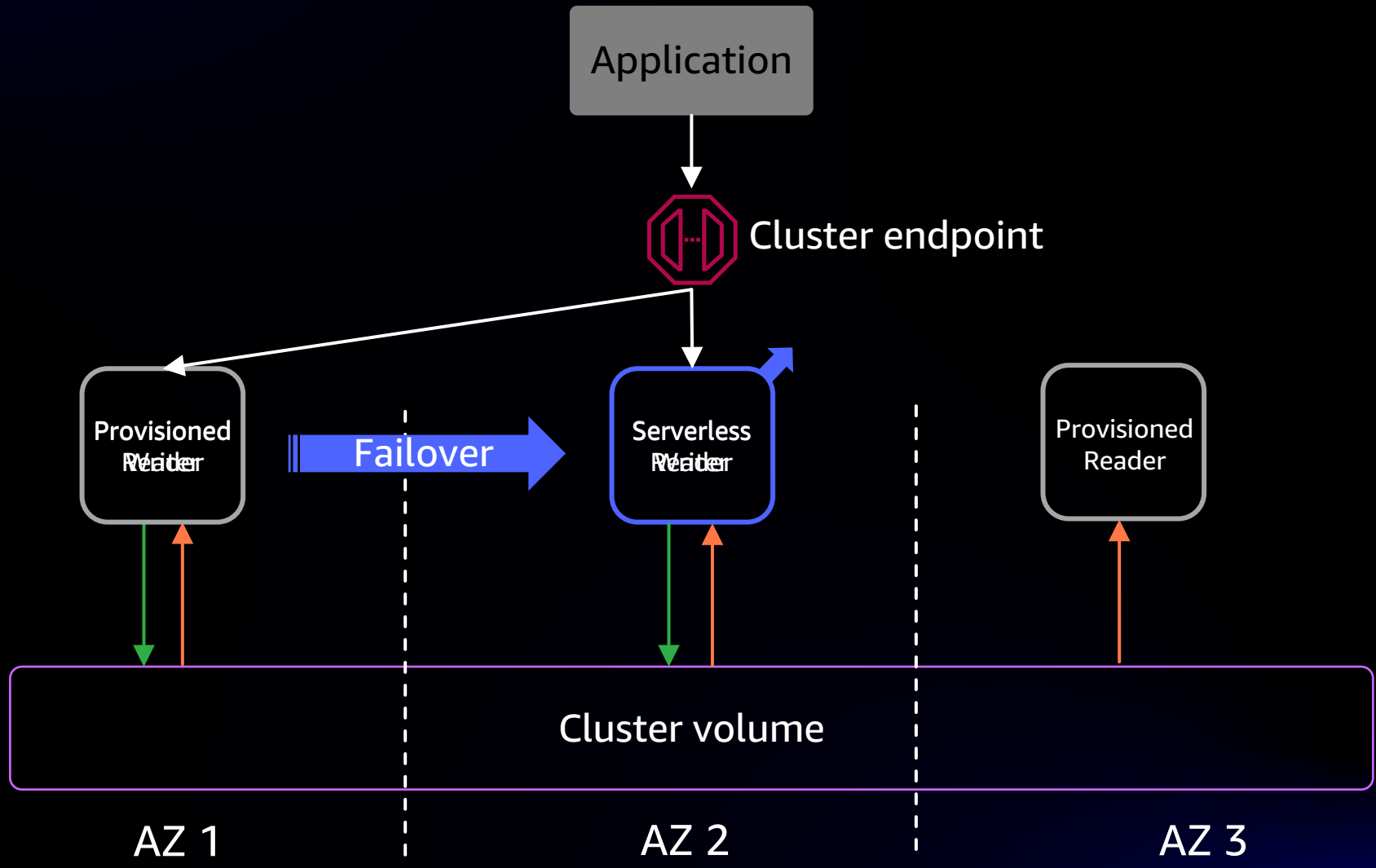
CloudWatch metrics

FREEABLEMEMORY (GIB) – AVAILABLE MEMORY



Getting started

Aurora clusters in mixed configuration



Demo: Mixed configuration of provisioned and serverless instances

Thank you!