

AWS RDS

CS516 – Cloud Computing

Computer Science Department

Maharishi International University

Maharishi International University - Fairfield, Iowa



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University.

Content

- Amazon RDS
 - DB instance and engine
 - Parameter group
 - Billing
- Amazon Aurora
 - Aurora DB cluster
 - Backups
 - Global table
 - Aurora serverless
 - Cloning
- RDS Read Replicas vs Aurora Read Replicas



Amazon Relational Database Service (RDS)

RDS is a web service that makes it easier to **set up, operate, and scale** a **relational database** in the AWS Cloud.

It provides cost-efficient, resizable capacity for an industry-standard relational databases and manages common database administration tasks.

To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also **restricts access** to certain system procedures and tables that require advanced privileges.

RDS benefits

- When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can **scale them independently**. If you need more CPU, less IOPS, or more storage, you can easily allocate them.
- Amazon RDS manages **backups**, software patching, automatic failure detection, and recovery.
- You can get high availability with a primary instance and a synchronous secondary instance that you can **fail over** to when problems occur. You can also create **read replicas** to increase read scaling and promote to a primary instance if needed.

DB instance and engine

- **DB instance** is an isolated database environment in the AWS Cloud. Your DB instance can contain multiple databases. You run DB instance in the **VPC** in private subnet. A **security group** controls the access to a DB instance.
- Each DB instance run **DB engine** such as MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server. Each DB engine has its own supported features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases.




RDS storage types

- **General Purpose SSD** – Offers cost-effective storage. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS. Baseline performance for these volumes is determined by the volume's size. Use it for development and testing environments.
- **Provisioned IOPS** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput. It is consistent. Use it for production.
- **Magnetic** – Legacy volume. AWS recommends that you use General Purpose SSD or Provisioned IOPS for any new storage needs.

Burst vs. Baseline Performance

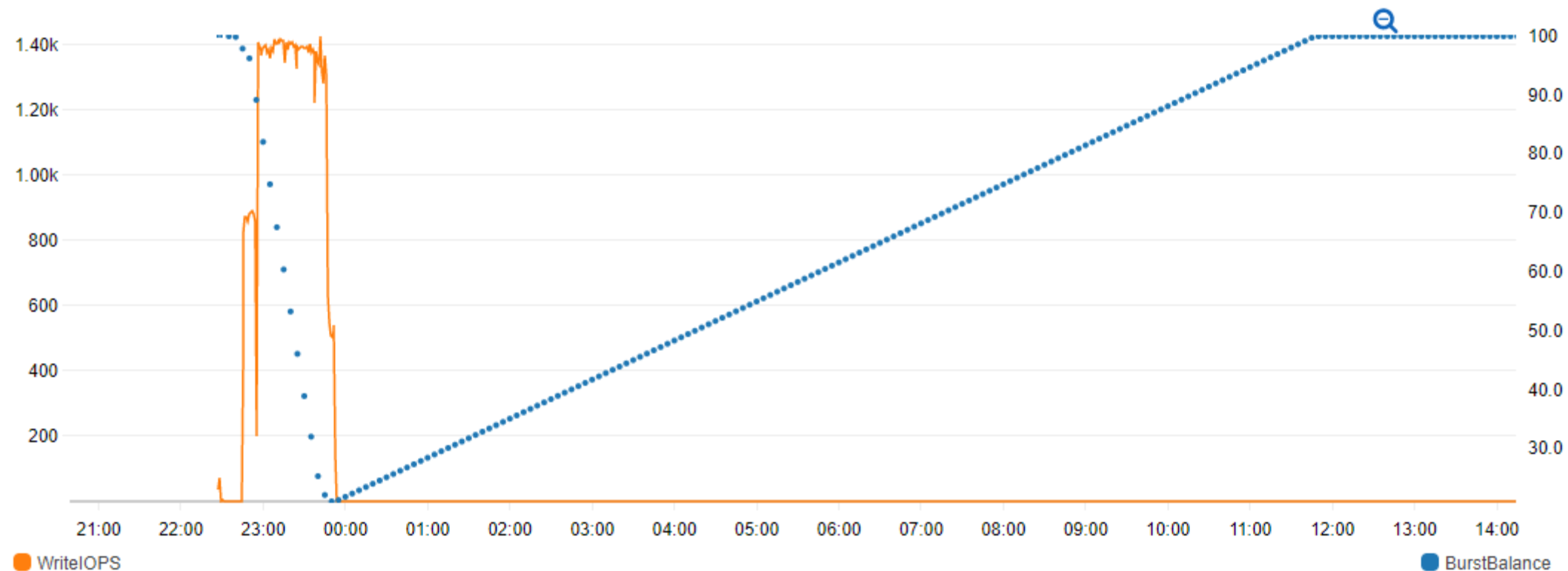
- The gp2 storage type has a **base IOPS** that is set when the volume is created. However, you don't provide a value for the IOPS directly—instead, IOPS is a function of the size of the volume.
- The IOPS for a gp2 volume is the size of the volume in GiB x 3, with a minimum of 100 IOPS and a maximum of 10K IOPS.
- To understand burst mode, you must be aware that every gp2 volume regardless of size starts with 5.4 million I/O credits at 3000 IOPS.
Works out to 3000 IOPS for 30 minutes. Then it is exhausted! There will be too much latency even if CPU and memory utilization is low.
- This is ideal for “bursty” workloads, such as daily reporting and recurring jobs.

burst versus writeiops 

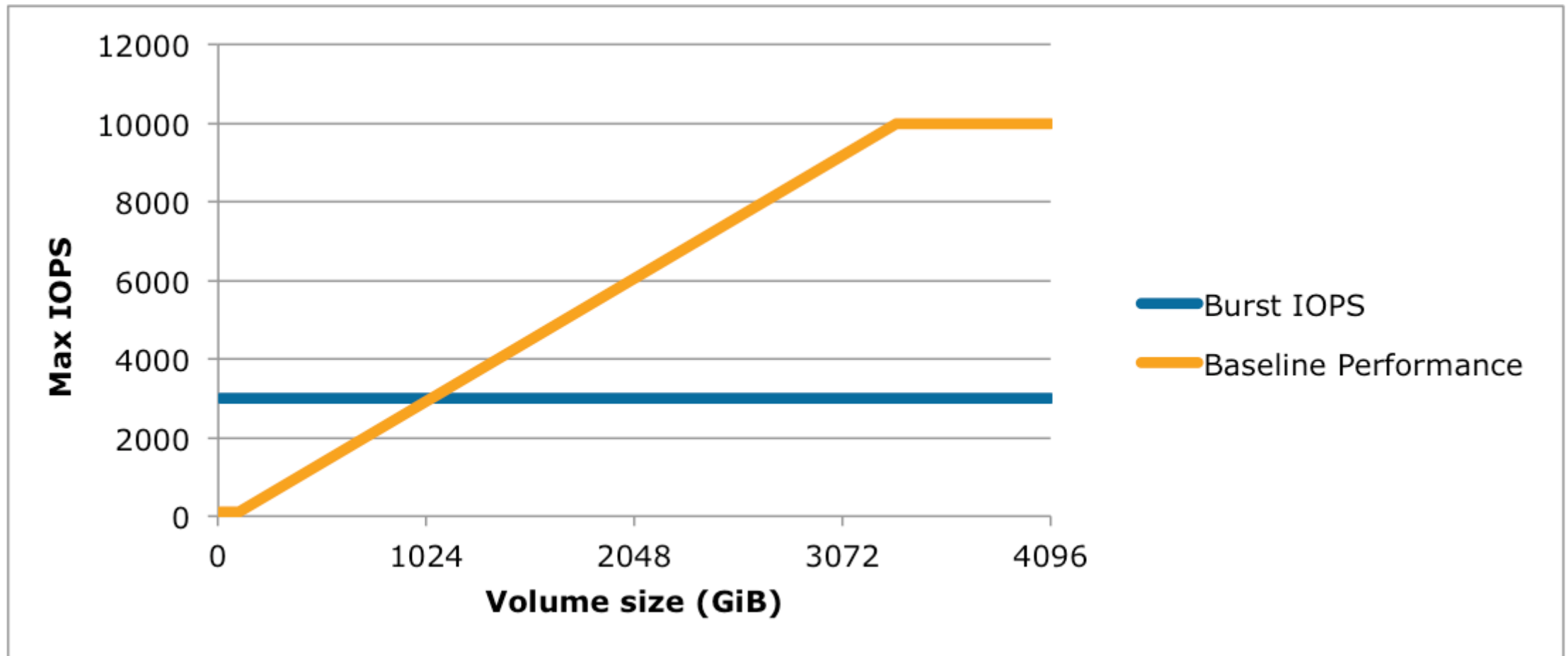
1h 3h 12h 1d 3d 1w custom (6d) ▾

Line ▾

Actions ▾



An important thing to note is that for any gp2 volume larger than 1 TiB, the baseline performance is greater than the burst performance. For such volumes, burst is irrelevant because the baseline performance is better than the 3,000 IOPS burst performance.



RDS parameter group

You manage your **DB engine configuration** by associating your DB instances with **parameter groups**.

Amazon RDS defines parameter groups with default settings that apply to newly created DB instances. You can't change the default parameter group. Instead create a new one.

Examples of using parameter group:

- Increase maximum number of connections.
- Enable BinLog to export data to data warehouse.
- Disable the foreign key constraint check at session layer.

You can't globally disable the foreign key constraint check. Because AWS manages RDS and has such restrictions to prevent someone accidentally blow up the entire DB.

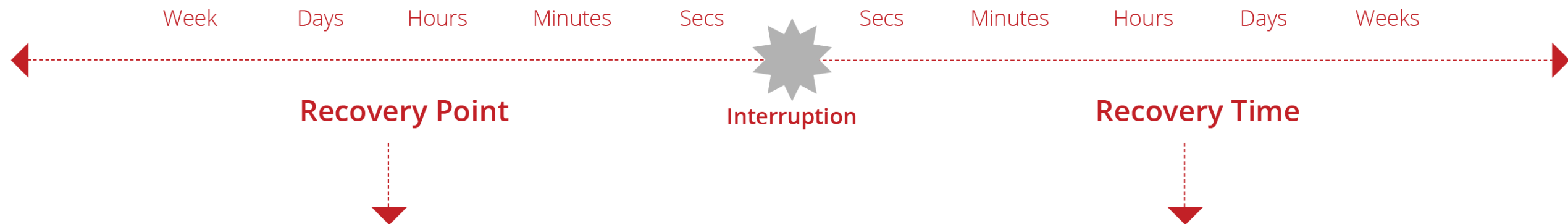
Instance classes and billing

RDS instance classes and billing model are similar to EC2. You have memory optimized, low latency and high IOPS, general purpose instance class types. For billing, you can choose either on-demand (expensive) or reserved (cheaper, 1 to 3 year commitment) instances.

You will be charged by:

- Per hour
- Storage
- I/O requests
- Provisioned IOPS
- Backup storage
- Data transfer

RPO vs. RTO



RECOVERY POINT OBJECTIVE

- Focuses on how you rebound from the loss of your data
- The interval of time between data backups and the loss of data
- Determines how often you should backup your data
- Considers how often your data changes

RECOVERY TIME OBJECTIVE

- Focuses on your business as a whole
- How fast you need to recover your data
- Determines how much preparation and budget you need to recover
- Considers how much downtime you can handle

RDS Backups and Snapshots

Backups are **automatically enabled** in RDS daily. The retention period is 7 days by default. You can decrease or increase it up to 35 days.

Transaction logs are backed-up by RDS every 5 minutes, which gives us the ability to **restore to any point in time**.

There is also the **continuous backup** and point-in-time recovery (PITR) feature. With this, database and backup administrators are able to reduce their recovery point objective (RPO) to 5 minutes or under. It is not supported in Amazon Aurora.

DB Snapshots are manually **triggered by the user**.

When you restore backups, you need to create and configure a new instance which takes approximately 45 minutes.

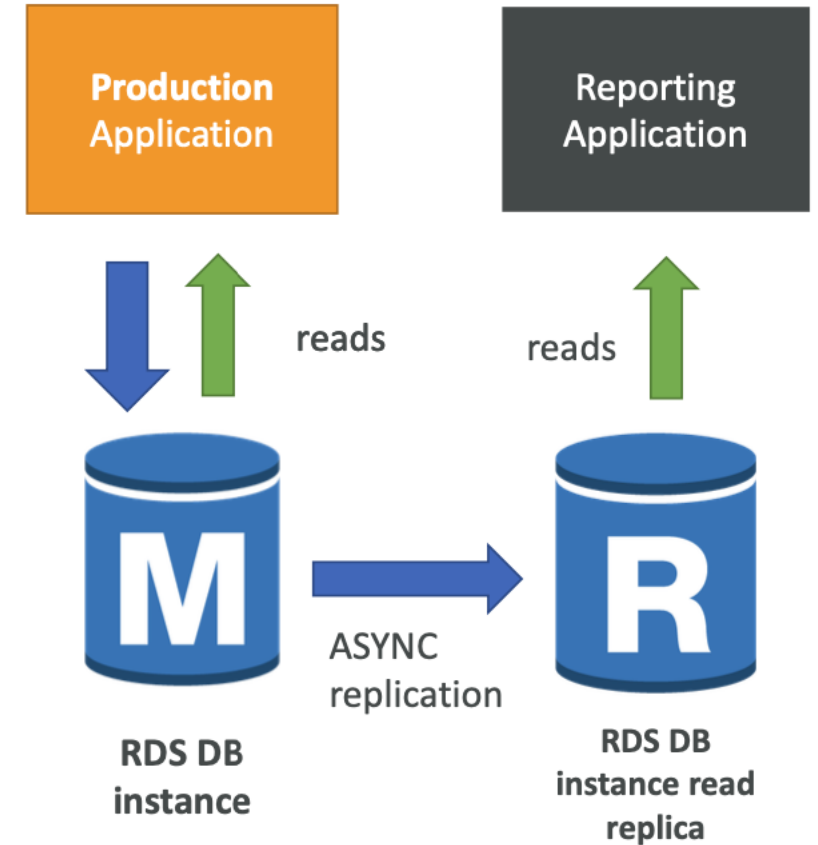
RDS Read Replicas

Amazon RDS Read Replicas provide

- enhanced performance
- scalability
- increased read throughput
- durability

Replication is ASYNC, so reads are eventually consistent. All transactions are secure.

Replicas can be promoted to their own DB. The Read Replicas be setup as **Multi AZ** for Disaster Recovery.



Multi-AZ deployment - RDS automatic failover

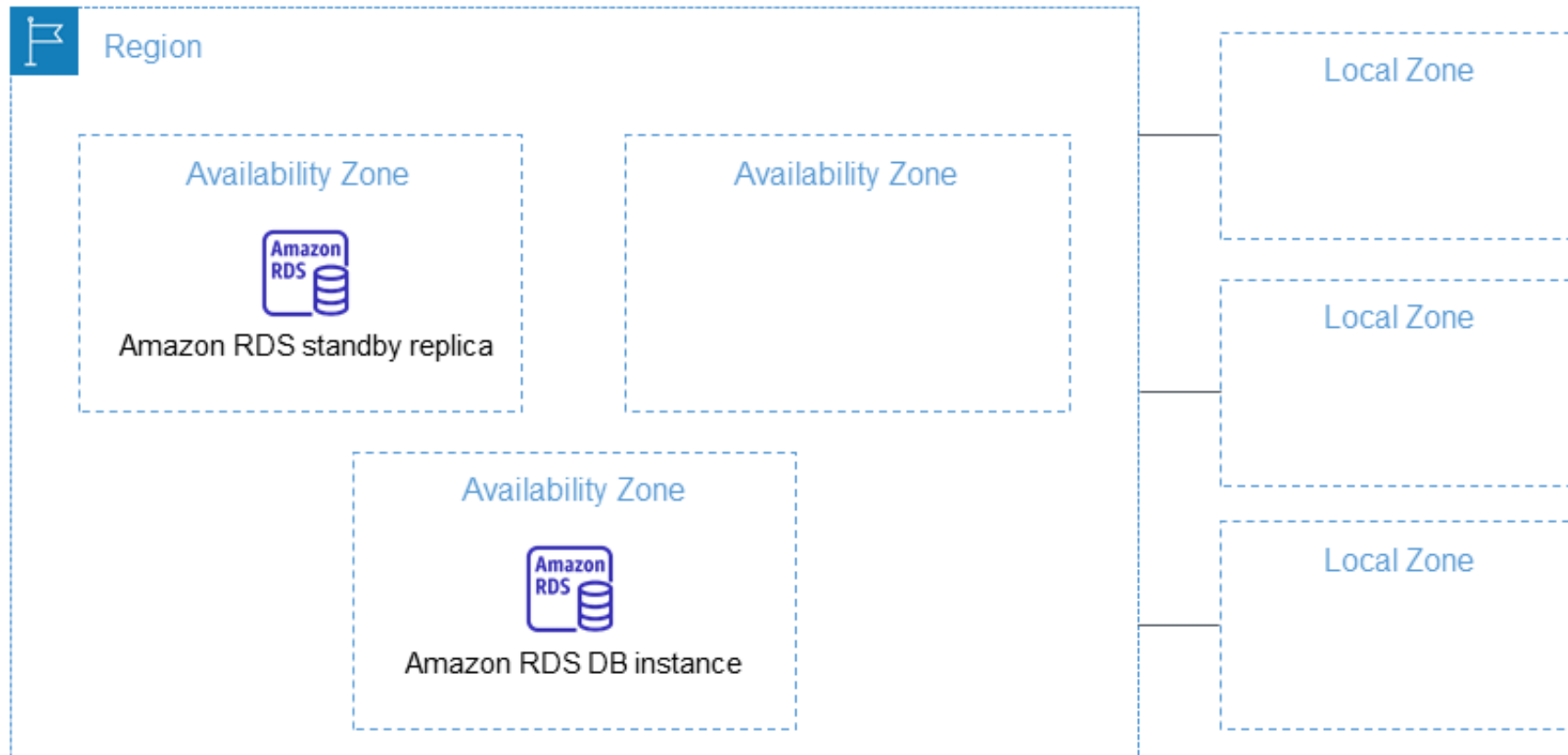
[In a Multi-AZ deployment](#), Amazon RDS automatically provisions and maintains a **synchronous standby** replica in a different Availability Zone. Automatically fails over in 1-2 minutes.

Standby replica provides

- data redundancy
- eliminate I/O freezes
- minimize latency spikes during system backups
- enhance availability during planned system maintenance

Multi-AZ deployment - RDS automatic failover

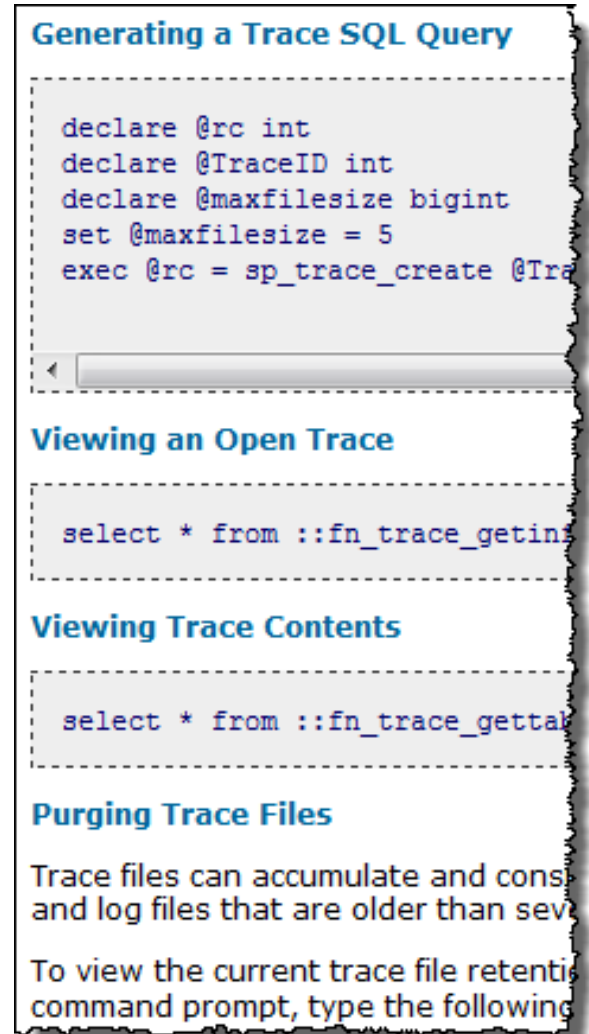
Only supported in MariaDB, MySQL, Oracle, and PostgreSQL DB. Uses the engine features Server Database Mirroring (DBM) or Always On Availability Groups (AGs).



Database Engine Tuning Advisor for Amazon RDS

The Amazon Relational Database Service for SQL Server supports the Microsoft SQL Server Database Engine Tuning Advisor.

The Advisor will help you to select and create an optimal set of indexes, indexed views, and partitions even if you don't have an expert-level understanding of the structure of your database or the internals of SQL Server.



RDS over a DB on EC2

- Automated provisioning, OS patching
- Continuous backups and restore to specific timestamp
- Read replicas for improved read performance
- Multi AZ setup for Disaster Recovery
- Scaling capability by creating read replicas

Amazon Aurora

AWS build Amazon Aurora on top of the open-source **MySQL** and **PostgreSQL**. It provides more features and is fully managed by AWS.

Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL. It takes advantage of fast **distributed storage**. The underlying storage grows automatically as needed.

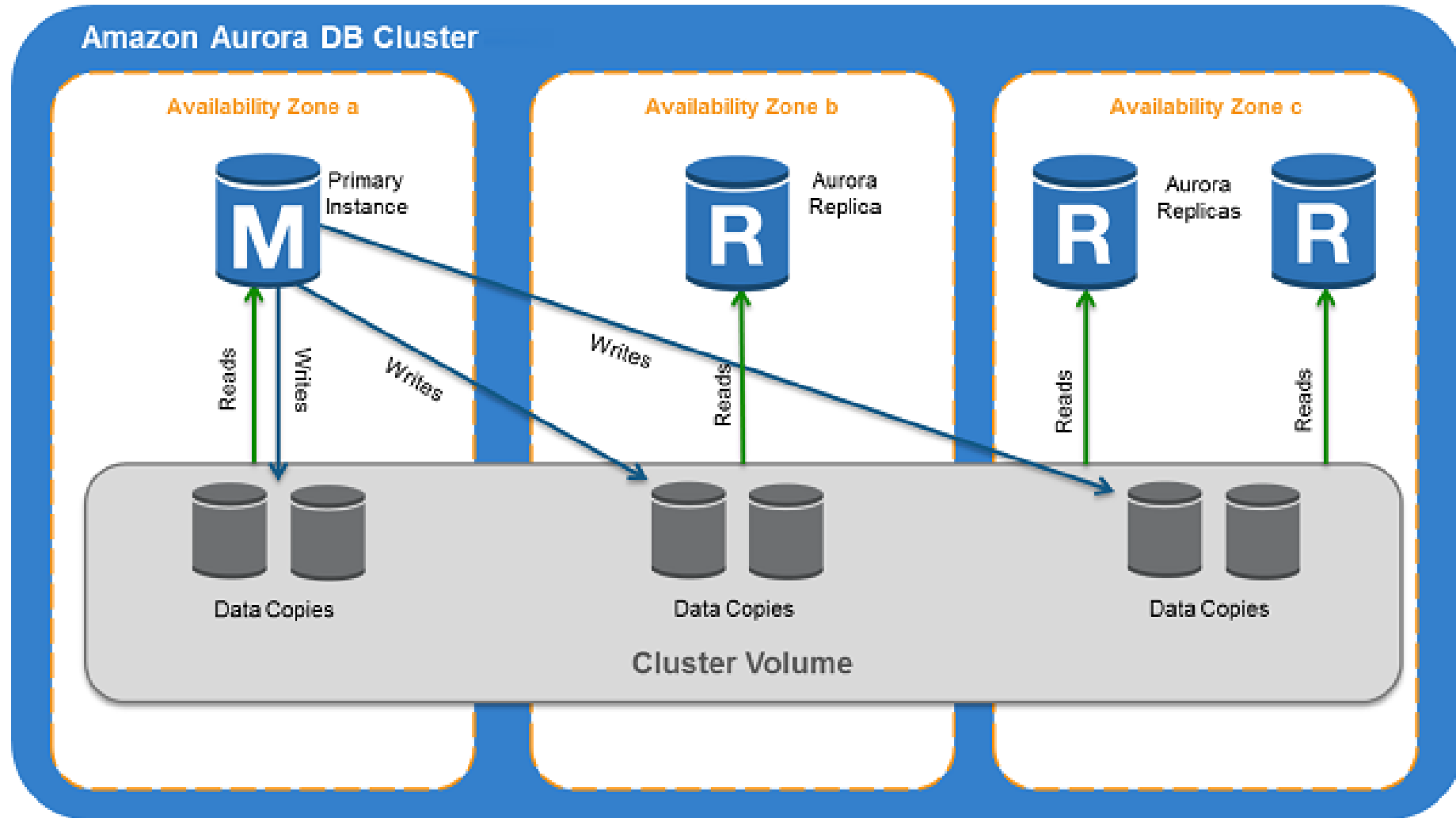
With Amazon Aurora, compute and storage is decoupled.

Amazon Aurora DB clusters

An Amazon Aurora DB cluster consists of one or more **DB instances** and a cluster **volume**. An Aurora **cluster volume** is a virtual database storage volume that spans **multiple AZs**, with each AZ having a copy (**replication**) of the DB cluster data. Two types of DB instances make up an Aurora DB cluster:

1. **Primary DB instance** – Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary DB instance.
2. **Aurora Replica** – Connects to the same storage volume as the primary DB instance and supports **only read** operations.

Amazon Aurora DB clusters



Amazon Aurora Features

Backtracking – you return the state of an Aurora cluster to a specific point in time, without restoring data from a backup. It completes within seconds, even for large databases.

Global databases – a single database that spans multiple AWS Regions, enabling low-latency global reads and disaster recovery from any Region-wide outage. It provides built-in fault tolerance.

Parallel queries – provides faster analytical queries over your current data while maintaining high throughput for your core transactional workload.

Machine learning – simple integration with ML services in AWS. You use standard SQL to build applications that call ML models, pass data to them, and return predictions as query results.

Survivable cache warming – Aurora caches common queries that improves DB performance.

Amazon Aurora Read Replica

Amazon Aurora replicas **share the same underlying storage** as the source instance, lowering costs and **avoiding the need to copy data** to the replica nodes.

Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary DB instance. Maintain **high availability** by locating Aurora Replicas in separate Availability Zones.

Aurora automatically **fails over** to an Aurora Replica in case the primary DB instance becomes unavailable.

Amazon Aurora Read Replica and Fail-Over

There is no sync or async in Amazon Aurora since it doesn't copy any data, instead just share the underlying storage. Replicas return the same data for query results with **minimal replica lag**. This lag is usually much less than 100 milliseconds after the primary instance has written an update.

If the primary instance in a DB cluster using single-master replication fails, Aurora automatically fails over to a new primary instance in one of two ways:

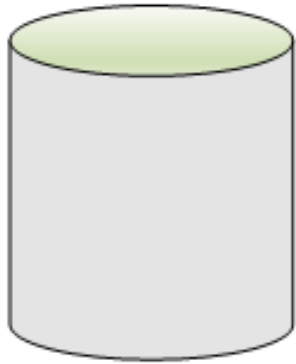
1. By promoting an existing Aurora Replica to the new primary instance (1 to 2 minutes)
2. By recreating a new primary instance if there isn't any replicas (10 minutes)

Cloning an Aurora DB cluster volume

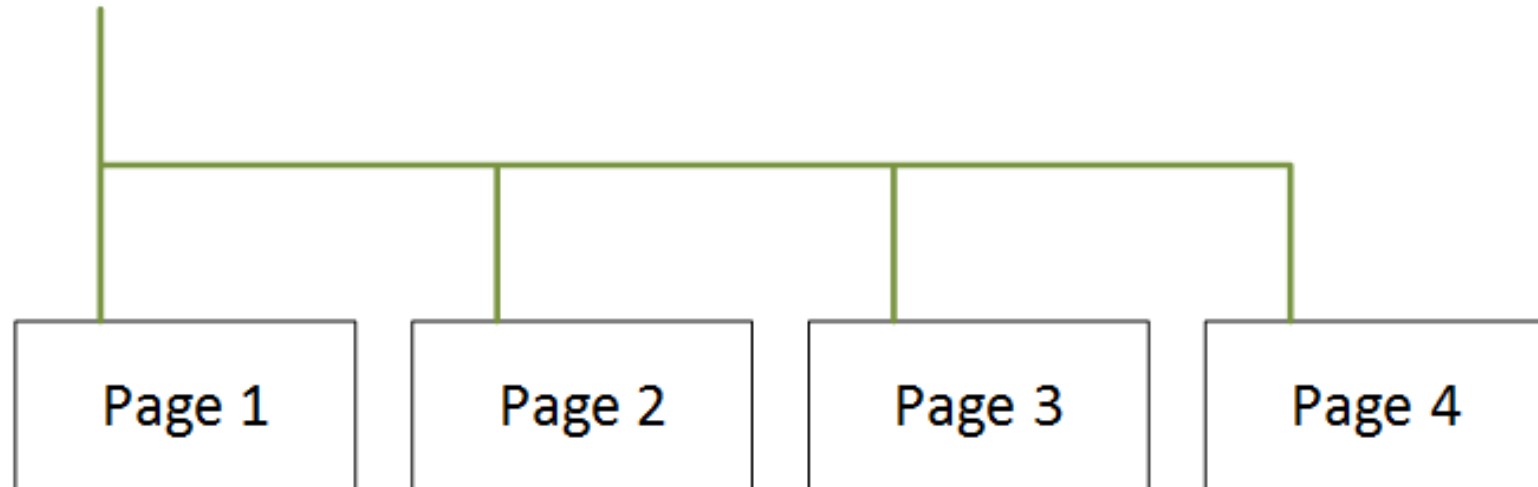
Using the Aurora cloning feature, you can quickly and cost-effectively create a new cluster containing a **duplicate of an Aurora cluster volume** and **all its data**. We refer to the new cluster and its associated cluster volume as a clone.

Creating a clone is faster and more space-efficient than physically copying the data using a different technique such as restoring a snapshot.

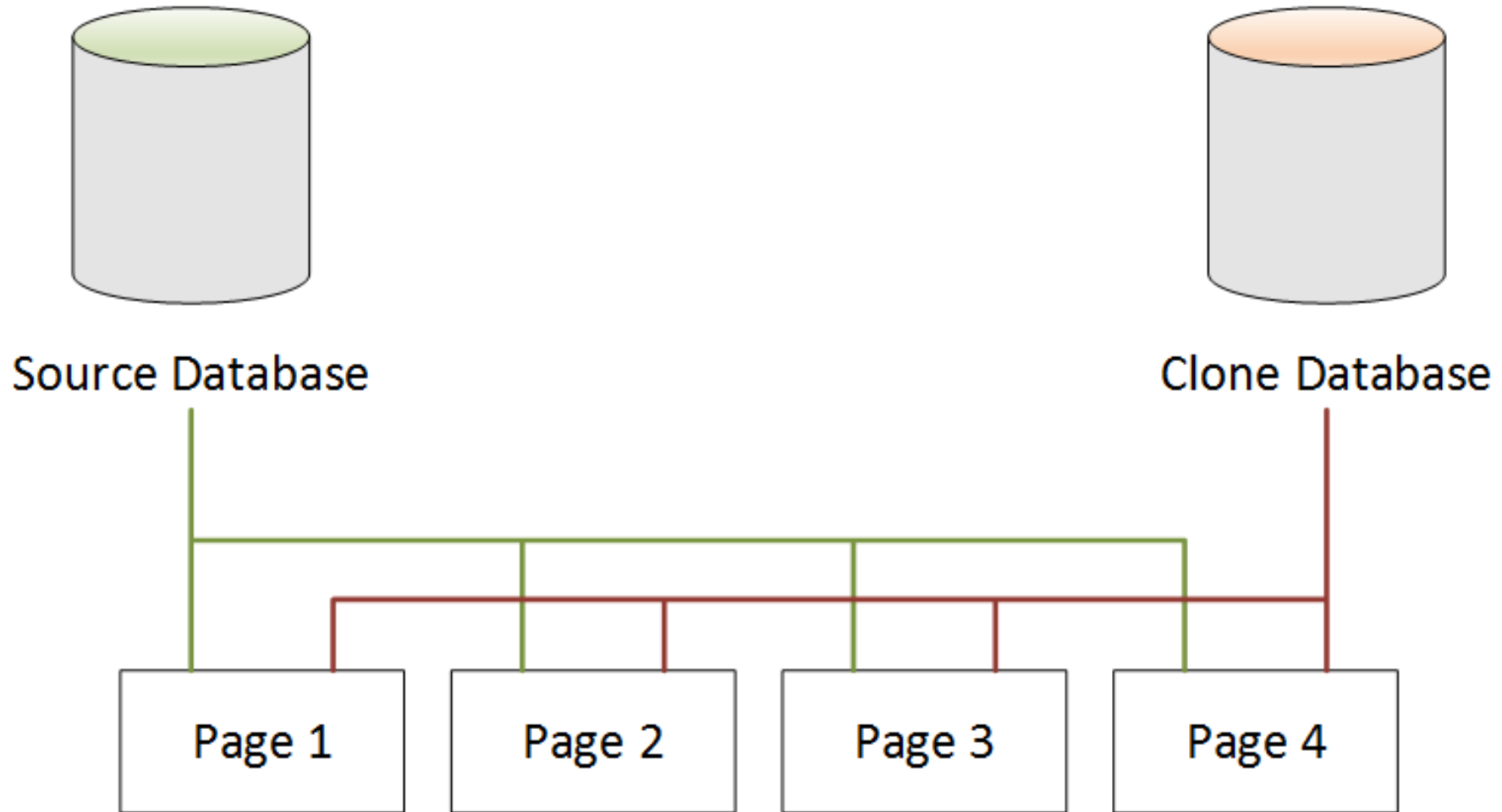
Before cloning



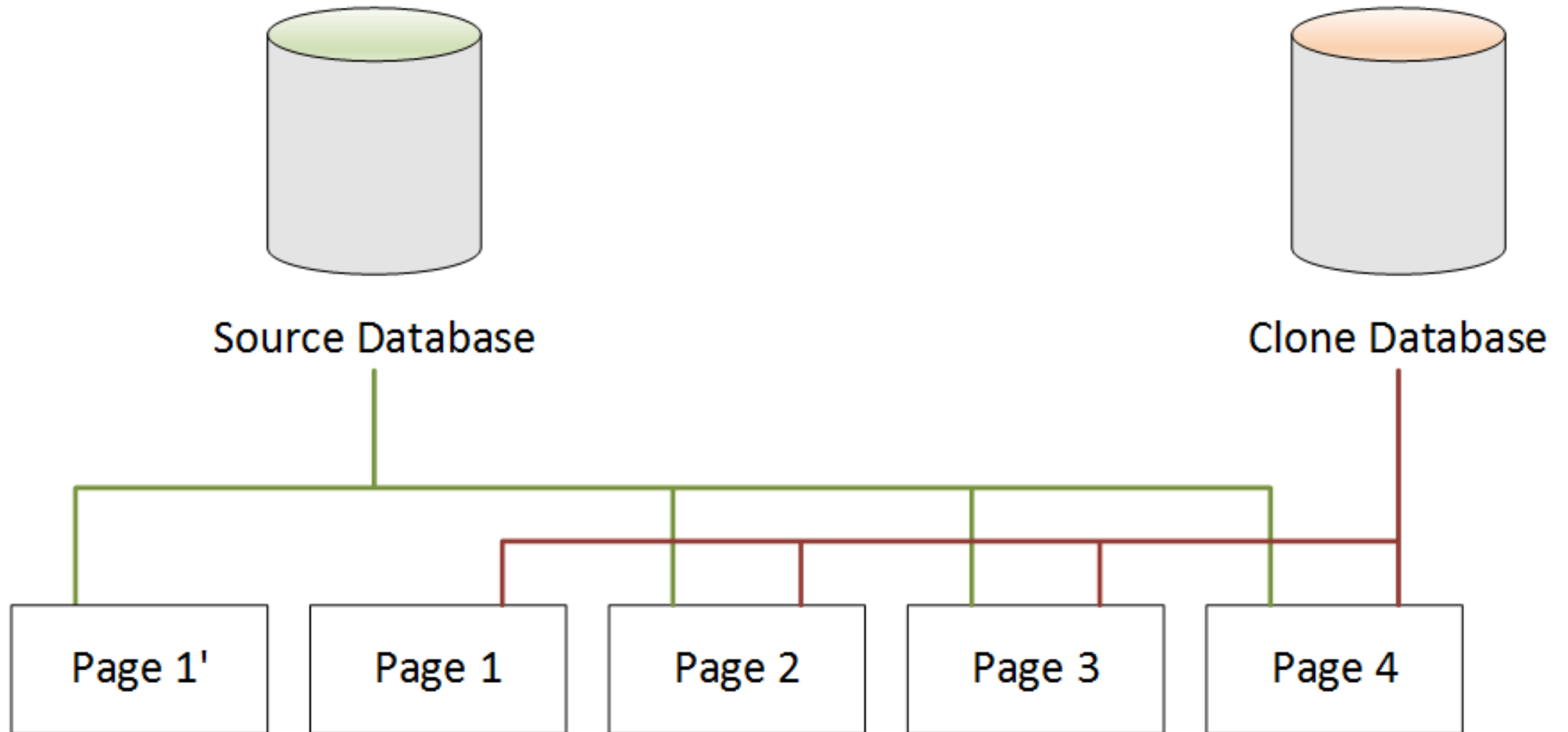
Source Database



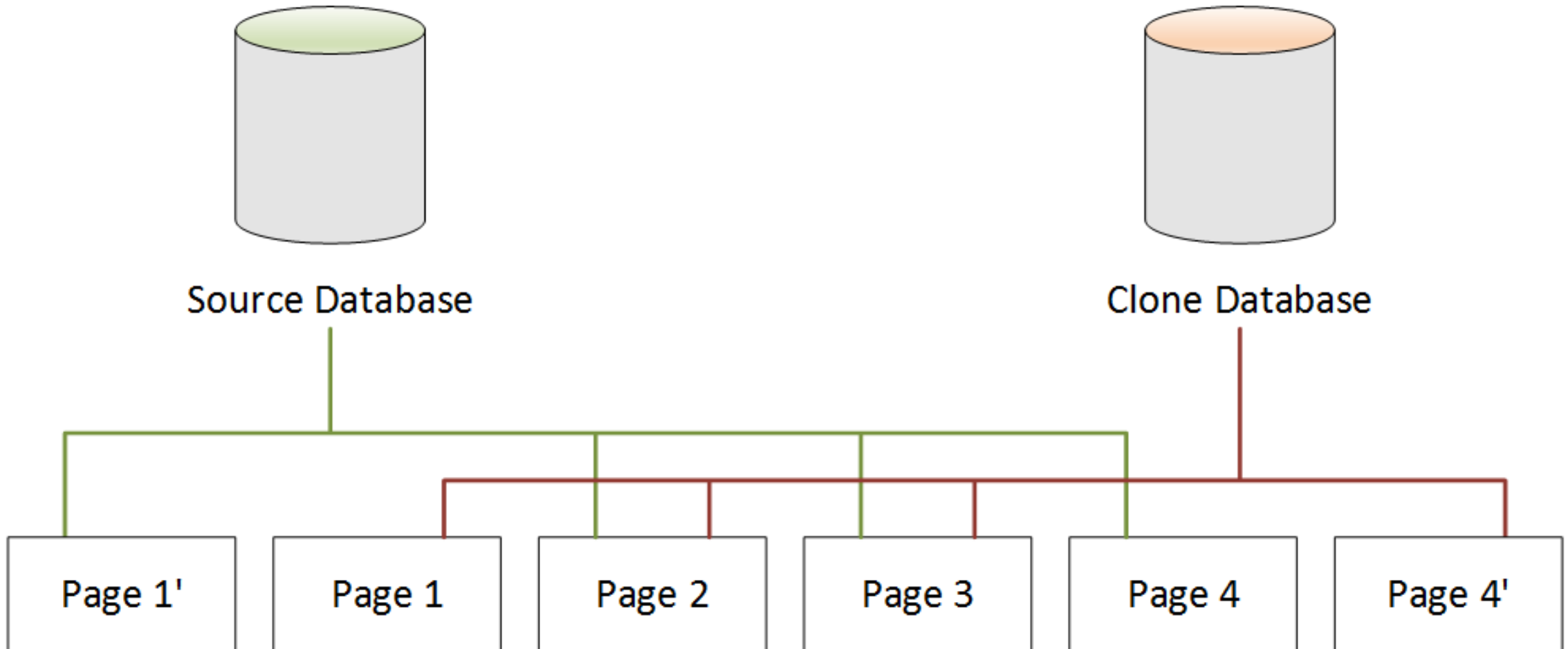
After cloning



When a change occurs on the source cluster volume



When a change occurs on the clone



Aurora serverless

Aurora serverless is an **on-demand autoscaling** configuration for Amazon Aurora. An Aurora Serverless DB cluster is a DB cluster that scales compute capacity up and down based on your application's needs.

This contrasts with Aurora provisioned DB clusters, for which you **manually** manage capacity.

Aurora Serverless is a relatively simple, cost-effective option for **infrequent, intermittent, or unpredictable** workloads.

Aurora serverless v2

Amazon Aurora Serverless v2 has been architected from the ground up to support serverless DB clusters that are **instantly scalable**. Scales up to hundreds-of-thousands of transactions in a fraction of a second.

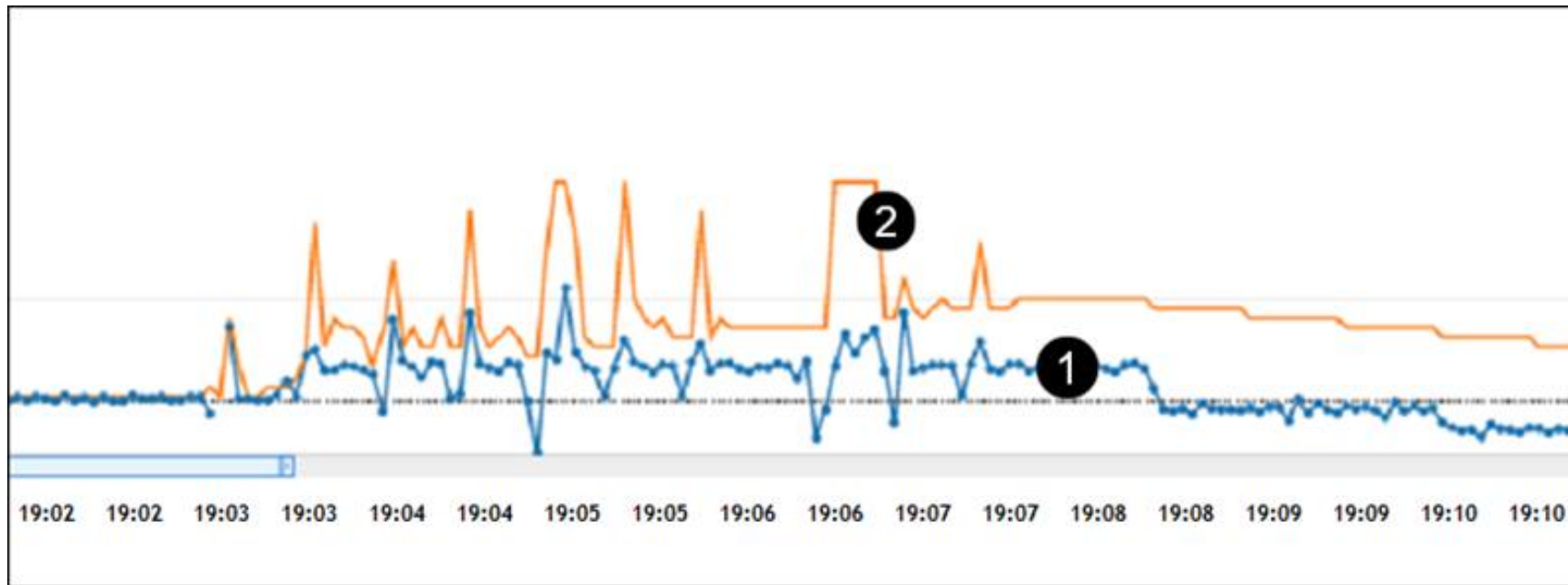
Aurora capacity units (ACUs) - One ACU provides 2 GiB (gibibytes) of memory (RAM) and associated virtual processor (vCPU) with networking.

Minimum ACUs (starts from 0.5 ACU) – The smallest number of ACUs down to which your Aurora Serverless v2 DB cluster can scale.

Maximum ACUs (Up to 256 ACUs) –The largest number of ACUs up to which your Aurora Serverless v2 DB cluster can scale.

Aurora serverless v2 scaling

Unlike Aurora Serverless v1, which scales by doubling ACUs each time the DB cluster reaches a threshold, Aurora Serverless v2 (preview) can increase ACUs incrementally.



1 - Orders processed each second, 2 - Aurora capacity units (ACUs) – Memory and CPU applied over time to increasing and decreasing demand.

Scaled instantly as expected. Full demo is [Aurora Serverless v2 – Instant scaling](#).