Cloud Databases

Fully Managed Database Services

Content Prepared By: Chandra Lingam, Cloud Wave LLC

amazon web services

AWS Cloud Database Offerings

- Relational Database Service (RDS)
- DynamoDB NoSQL Database Service
- Redshift Data warehouse Service
- ElastiCache in-memory cache Service
- Database Migration Service

Relational Databases

- Data Stored as Tables, Columns and Relations
- Rigid Schema
- Usually normalized to minimize duplication
- Very popular with widespread usage
- SQL Query Language
- Scale-up processing

Movie Example

Movie Table

ID	Title	Year	Date Released
1	Rush	2013	2013-09-02
2	Prisoners	2013	2013-08-30

Actor Table

ID	Actor Name
100	Daniel Bruhl
101	Chris Hemsworth

Genre Table

ID	Genre
1	Action
2	Biography
3	Drama

Movie_Actor Table

Movie ID	Actor ID
1	100
1	101

Movie_Genre Table

Movie ID	Genre ID
1	1
1	3

Complex Query, Transaction, Typically used for “Write” heavy data

amazon web services

Movie – Denormalized Example

Movie Table

ID	Title	Year	Date Released	Actor_1	Actor_2	Genre_1	Genre_2
1	Rush	2013	2013-09-02	Daniel B..	Chris H..	Action	Drama
2	Prisoners	2013	2013-08-30	Hugh J..	Viola D..	Crime	Drama

Easy to Query, More Storage, Duplication, Typically used for “Read-Only” Data

NoSQL Databases

- High Performance, Non-relational databases
- Variety of data storage
  - Key-Value Pairs
  - Document Store
  - Graph
  - Column Oriented
- Denormalized, Easy to change “Schema”
- Some support SQL Query Language
- Scale-out processing

NoSQL Document Movie Example

Review: OneMovie.json Example File

Reality: NoSQL often “normalized” into separate tables to optimize for specific use case. For example: Movie in one table, Movie Reviews in another table, Ratings in third table and so forth

Columnar Databases

- Optimized for reading and writing specific columns
- Ideal for analytic query performance – when analyzing select columns
- Reduced Disk I/O requirement
- High level of compression
- Scale-out processing
- Example: Redshift Columnar Storage

In-Memory Storage

- In-memory database optimized for read-heavy workloads and compute intensive workloads
- Extremely fast sub-millisecond response time
- Reduce traffic hitting backend databases
- Ideal for social networking, gaming leadership boards, API request limit throttling, recommendation engines

AWS ElasticSearch Service

- Text Search Engine
- Faceting Search allows categorization of search results
    - Price Range
    - Product Types (Video, Book, Toys)
    - Brands
- Suggestions as you type
- Reduce traffic hitting backend databases
- Very fast response times
- Used for application search, log analytics, application monitoring, clickstream analytics

Relational Versus NoSQL

Table: Comparison Between Relational and NoSQL

Terminology: Relational and NoSQL

Relational Database Service (RDS)

Relational Database Service

- Six Popular Database Engines: Aurora, MySQL, MariaDB, PostgreSQL, Oracle, SQL Server
- RDS handles:
  - Server, Storage Provisioning – No instance admin access
  - Patching
  - Backup
  - Recovery
  - Failure Detection and Repair
  - Synchronous Replication with Multi-AZ high availability
  - Read Replicas

Benefits

- Lower Administrative Burden
- Performance
- Scalability
- Availability and Durability
- Security
- Manageability
- Cost Effective
- Choice of Instance Classes, Storage Options

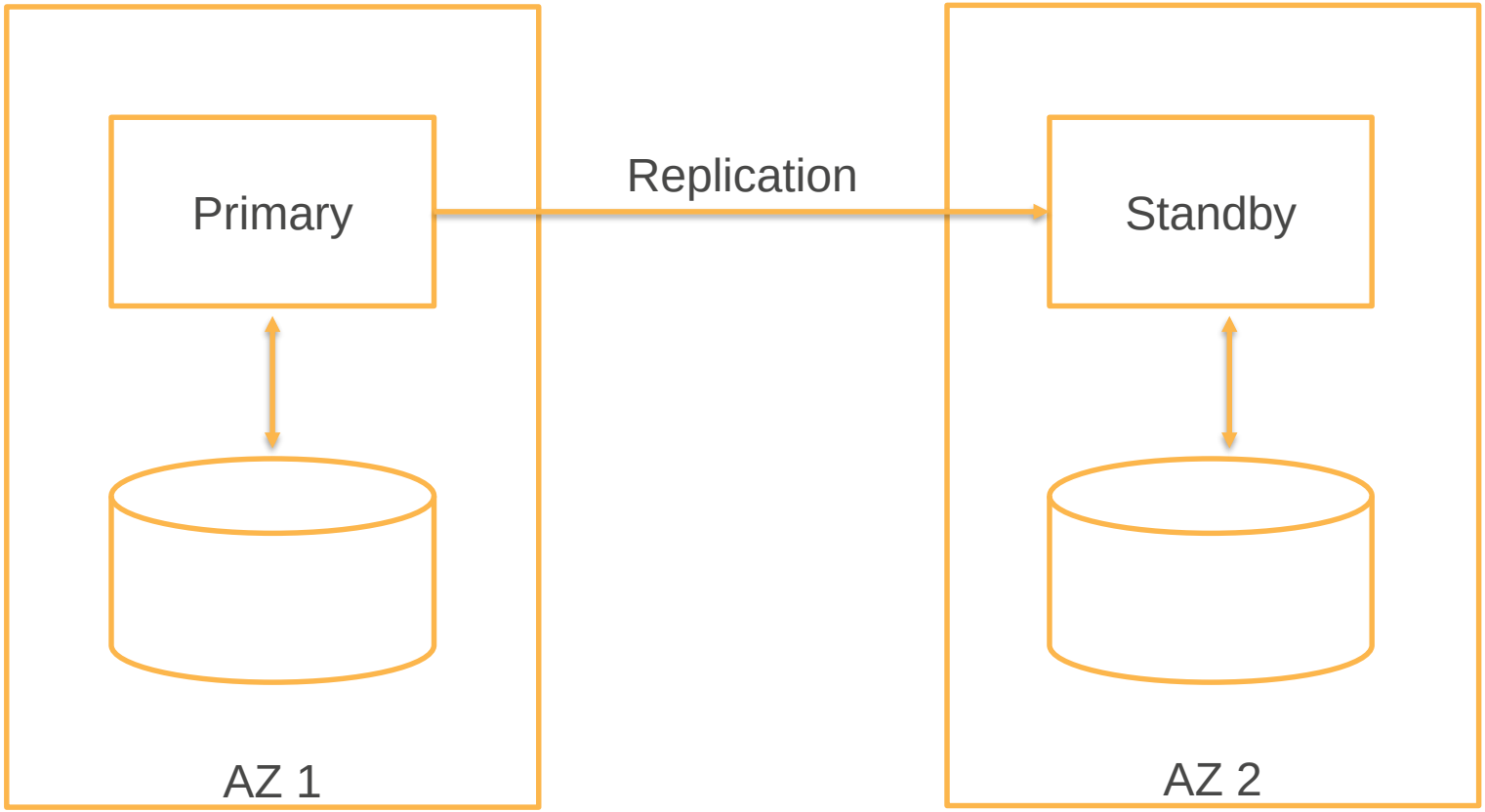
RDS Concepts/Terminology

- DB Instance – A database server with your choice of DB Engine.
- EBS Storage - 5GB to 6 TB (depends on DB engine)
- Databases – Each DB Instance contains one or more customer defined databases
- VPC – Configure DB Instance to run inside your VPC
- Multi-AZ Deployment – Maintains Primary and Stand-by instances in two separate AZs with synchronous replication

RDS Concepts

- Primary Instance – Handles all application traffic for read/write
- Standby Instance – Synchronous copy of Primary used for redundancy, failover support, system backups
- Security Group – Control access to DB Instance
- DB Parameter Group – Customize DB Engine and apply to one or more DB instances of same type
- DB Option Group – Optional features – memcache with MySQL, Data Encryption with SQL Server

Relational Database Engines – High Availability



Aurora

Overview

- MySQL and PostgreSQL compatible
- Performance: 5X MySQL, 3X PostgreSQL
- Six way data replication across 3 Azs
    - For Writes, four copies must be stored safely before transaction is successful
- In case of primary crash, a read replica is promoted as primary – typically under 60 seconds
- Low latency read replicas (up to 15)
- Support for Cross Region Replication

Aurora

- Cluster Endpoint
    - Points to Current Primary Instance
    - Suitable for Writes and Reads

mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306

- Reader Endpoint
    - Points to Read Replicas
    - Suitable for Reads
    - Multiple Read Replicas are load balanced at connection level

mydbcluster.cluster-ro-123456789012.us-east-1.rds.amazonaws.com:3306

- Instance Endpoint
    - Points to Individual Aurora Instance

Aurora Serverless (GA as of Aug 2018)

- Aurora Serverless - Suitable for use cases that are intermittent or unpredictable
- Specify Minimum, Maximum Aurora Capacity Units (ACU)
- 1 ACU is ~2 GB of Memory with corresponding CPU/Network
- Pricing 1 ACU is $0.06 per hour + Storage + I/O
- Aurora Serverless automatically scales up and down based on load
- Scaling is rapid – uses a pool of warm resources

Aurora Serverless

- Storage and Processing are separate – scale down to zero processing and pay only for storage
- Automatic Pause and Resume – Configurable period of inactivity after which DB Cluster is Paused
    - Default is 5 minutes
    - When paused, you are charged only for Storage
    - Automatically Resumes when new database connections are requested

Pricing

- Instance Class – On-demand, Reserved
- Storage – General, Provisioned
- Backup Storage
  - First 100% provisioned storage is free
- I/O Requests per month
- Data Transfer

Demo – RDS VPC, Client Preparation

- Setup VPC
- Setup Subnet Group
- Setup Security Groups
- Setup EC2 instance with MySQL Client
- Setup MySQL Workbench (Visual Editor)

Demo – MySQL Multi-AZ Deployment

- Setup MySQL Database with multi-AZ option
- Note: Multi-AZ deployment is not part of free-tier, so there will small charge. We will be using micro instances.
- Connect to Primary Instance using DNS Host Name
- Create Table, Add Data
- Demonstrate Fail-Over (primary down, standby is the new primary)
- Connect to new Primary and Query

Demo – Read Replica

Objective: Direct Read only traffic to a replica

- Create a new Read Replica
- Connect to Primary to Add new data
- Connect to Read Replica and Query

Demo – Backup and Snapshot

- Automated Backup
- Manual Backup
- Restoring backup to a new Instance

Instance Lifecycle

- Maintenance and Upgrades
  - Weekly Maintenance Windows
  - RDS can automatically apply patches to DB Instance or OS
  - Manually apply pending patches
  - Defer maintenance items
  - Maintenance Items marked as “Required” cannot be deferred indefinitely
  - Multi-AZ deployment: Maintenance is performed on standby, promote to primary and perform maintenance on old primary

Modifying Instance

Modifying Existing DB Instance – What can you change?

- Apply Immediately – to apply changes right away
- Defer to next scheduled maintenance window

Security

- IAM Control for RDS Resources

Note: You cannot connect to DB Instance with IAM Users. DB requires separate account

- Encryption
    - Data at rest
    - Automated Backup
    - Snapshot
    - Read Replicas
- SSL Connection

Customization

Option Groups – Manage optional features

Examples: SQL Server Enable DB Engine level encryption for data files, Memcached support for MySQL

DB Parameter Groups – Manage Engine Configuration

RDS Provides a default configuration based on DB Engine, Instance Class, Allocated Storage

Monitoring

CloudWatch Metrics for RDS Instances

Enhanced Monitoring

Enhanced Montioring Agent running on DB Instance measures metrics like CPU

CloudWatch metrics gathers CPU metrics from hypervisor

Architectural Best Practices

AWS Summit Series 2016 | Chicago - Big Data Architectural Patterns and Best Practices on AWS

AWS re:Invent 2016: ElastiCache Deep Dive: Best Practices and Usage Patterns (DAT306)

AWS Programmatic Integration

AWS Application Integration

- Setup Python Environment
- Install Boto3 AWS SDK
- Interact with AWS using Python

SDKs in several supported languages

Demo – Python Boto3

Objective: Demonstrate how to programmatically connect to AWS services

1. Establish session with user credentials – same credential that was configured with Command Line interface
2. Query S3 for list of buckets
3. In DynamoDB Demo, we will use Boto3 to manage Tables, Load, Query Data

DynamoDB

DynamoDB

- Fully Managed NoSQL Database Service
- Partitions - Store and Retrieve any amount of data
- Automatic deletion of expired items – Time To Live
- Automatic replication of data across all AZs in a region
- Automatic scaling based on customer specified Read and Write Capacity
- Read Consistency (Eventual, Strong)
- Downloadable Developer Version

DynamoDB Core Concepts

Tables

- Items
- Attributes

Primary Key

- Partition Key
- Partition Key and Sort Key

Secondary Indexes

- Global Secondary Index
- Local Secondary Index

DynamoDB Core Concepts

DynamoDB Streams

- Captures data modification events in DynamoDB Tables
- Ordered Set of events
- Near real-time
- Lifetime of 24 hours
- Figure: Lambda functions to process stream events

Examples: New customer – welcome email, Add new product to ElastiCache or ElasticSearch

Provisioned Throughput

- Consistent Low Latency Performance
- Read Capacity Units
- Write Capacity Units
- Modify any time
- Reduce cost – Purchase Reserved Capacity

Pricing

DynamoDB Pricing

- Provisioned Read/Write Capacity
- Index Storage
- DynamoDB Streams
- Capacity Reservation (lower cost with 1 year commitment)
- Data Transfer

Demo – Movie Database

- DynamoDB interaction using Python
- Create Tables, Load, Query Data
- Movie demos in other programming languages are available here
- Sample Movie Data

Monitoring

CloudWatch Metrics

Access from Table Metrics or through CloudWatch