AWS Databases

Relational, NoSQL, In-Memory, Data warehouse, Specialized

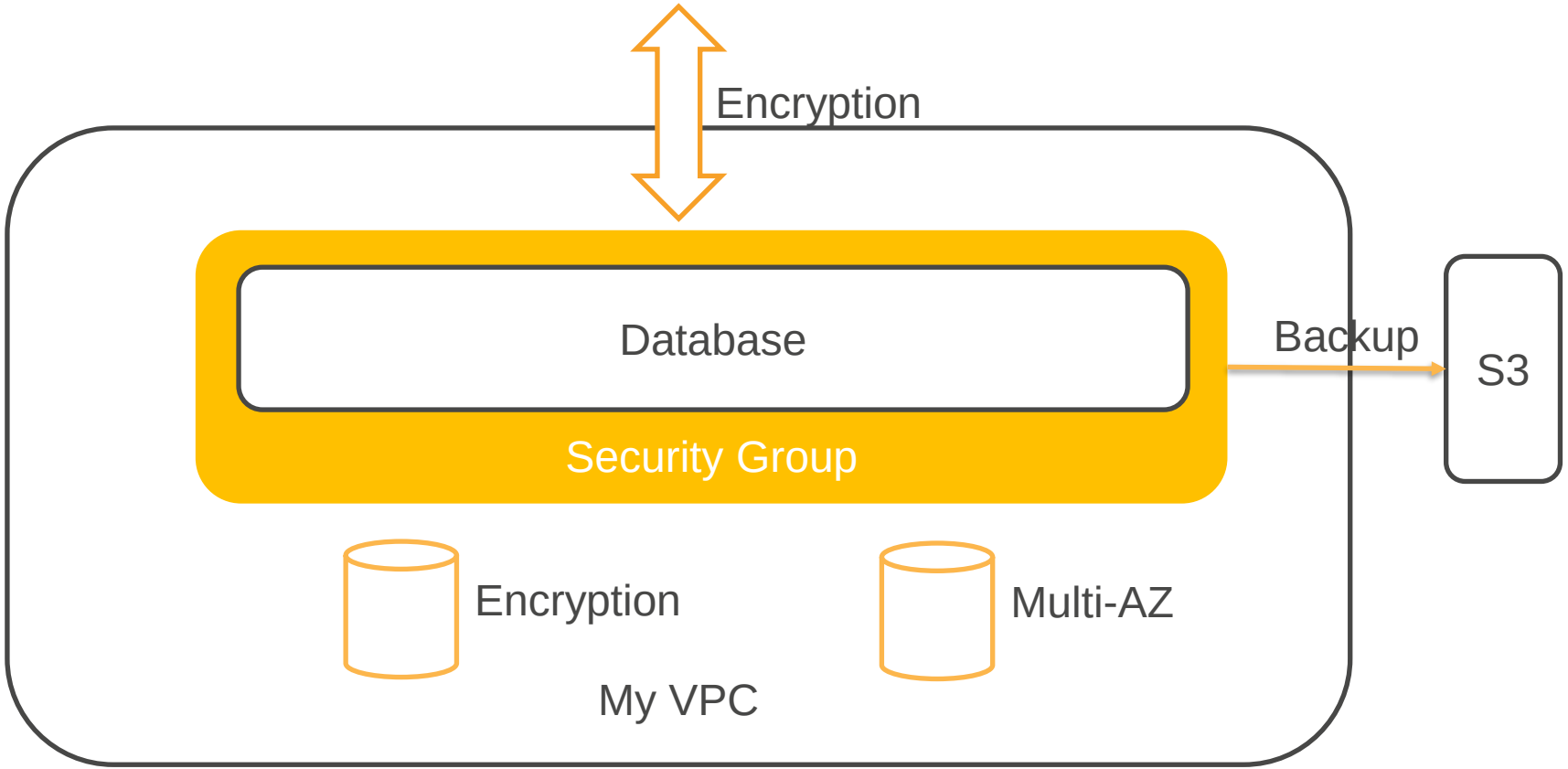
Chandra Lingam

Cloud Wave LLC

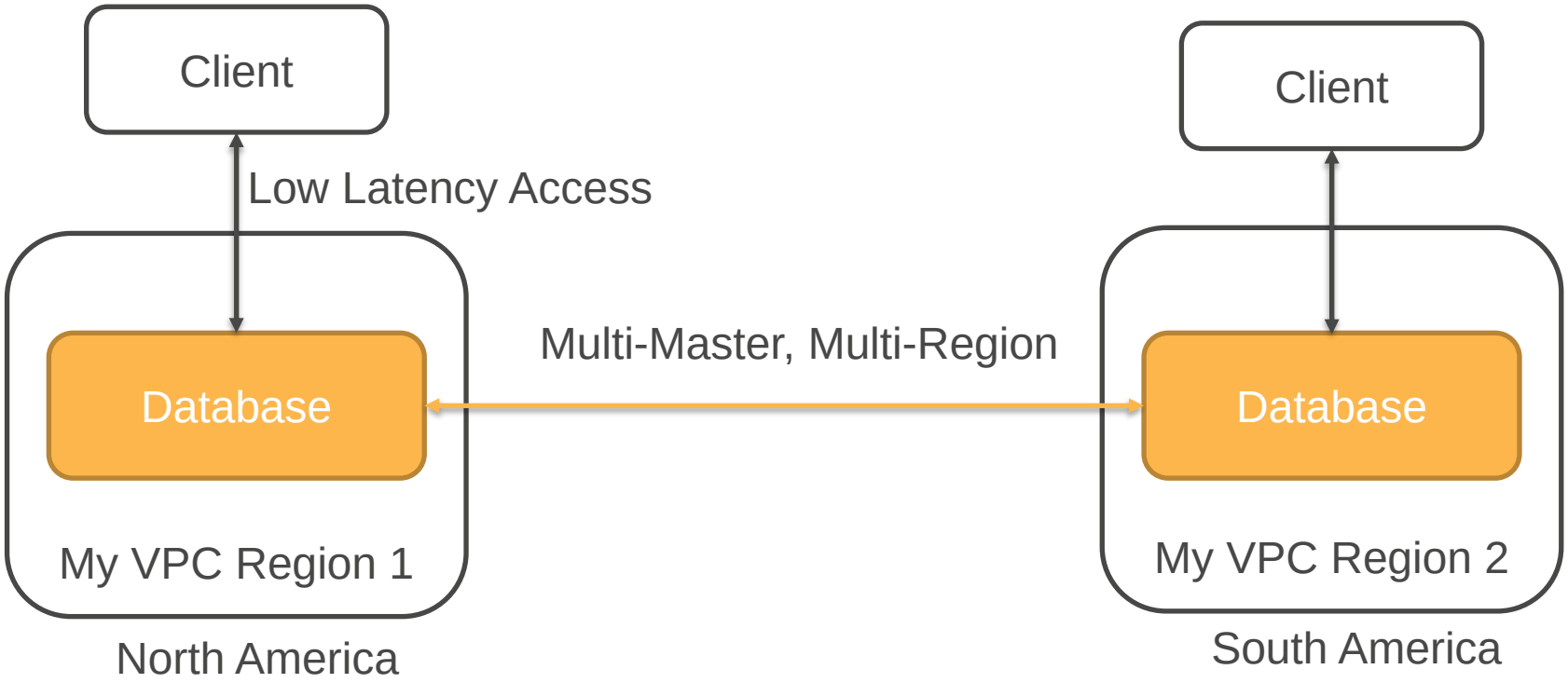
AWS Databases

Note: Not complete list

Standard Features – AWS Databases



DynamoDB Global Table - Multi-Region, Multi-Master



Benefits

- Wide selection of database engines
- Fully managed
- VPC Network Isolation
- Encryption at rest using KMS
- Encryption in transit
- Automated Backup
- Highly Durable and Available – Replicated across multiple devices in Availability Zone, Region
- Multi-Region, Multi-Master (some products) – Low latency access and disaster recovery

AWS Portfolio of Databases (1 of 2)

Service	Type of Database
RDS - Relational Database Service	Relational Database. Choice of database engines - Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, SQL Server Uses: Traditional applications, ERP, CRM, ecommerce
Redshift	Petabyte scale Data warehouse, Massively Parallel Columnar Storage, integrates with S3 data lake Uses: business intelligence, analytics, SQL to explore data lake
DynamoDB, Cassandra, DocumentDB	NoSQL Database Key-value storage, document store, consistent single digit millisecond latency at any scale Uses: high traffic web applications, ecommerce, gaming systems
ElastiCache	In-memory database - MemCached, Redis Sub-millisecond latency Uses: Caching, user session, gaming leaderboards, geospatial applications

AWS Portfolio of Databases (2 of 2)

Service	Type of Database
Neptune	Graph Database – optimized for highly connected datasets and querying relationships Uses: Social networks, recommendation engines
Timestream	Timeseries Database – optimized for storing and querying high volume timeseries data at 1/10th the cost of relational databases Uses: IoT applications, Industrial telemetry, DevOps
Quantum Ledger Database	Ledger Database – Blockchain based system for transparent, immutable, and cryptographically verifiable transaction log Uses: Systems of record, supply chain, banking transactions
Elasticsearch	Search database, store, analyze and correlate logs from disparate applications and systems Uses: search, infrastructure and application monitoring, Security info and event management

Database Migration

AWS Database Migration Service (DMS)

- One-time data replication
- Continuous data replication from on-premises to AWS (and reverse)
- Homogeneous and Heterogeneous replication

Summary

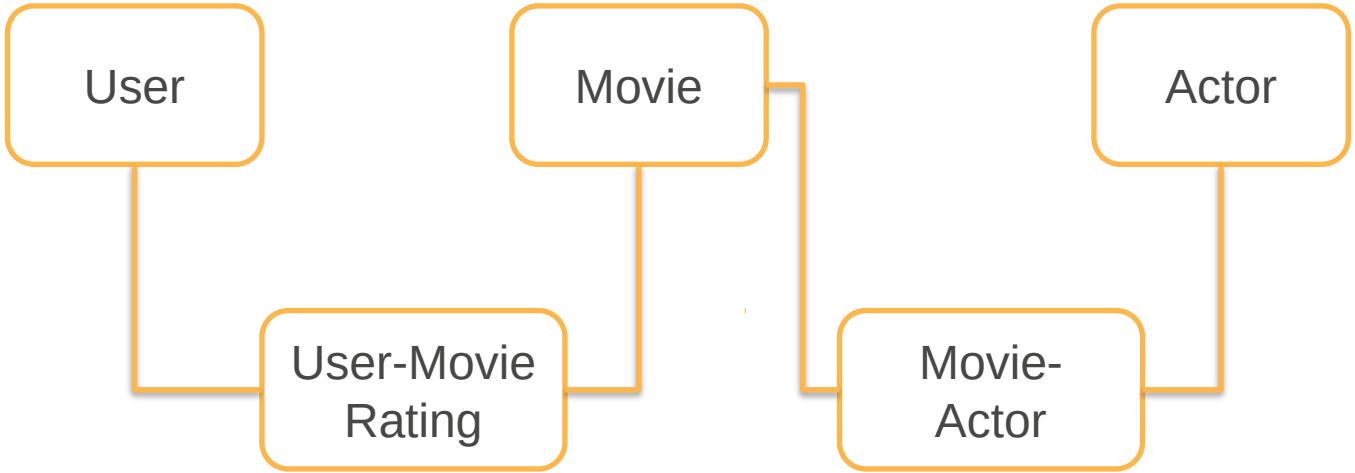
"The broadest selection of purpose-built databases for all your application needs"

"By picking the best database to solve a specific problem or a group of problems, you can breakaway from restrictive one-size-fits-all monolithic databases“

Reference: https://aws.amazon.com/products/databases/

Relational Database Service (RDS)

Relational Database



Relational Database

General Purpose – Design a schema for any need

Rigid Schema – difficult to change

SQL – Flexible Querying System

Complex System

Scaling Challenges

Amazon Relational Database Service (RDS)

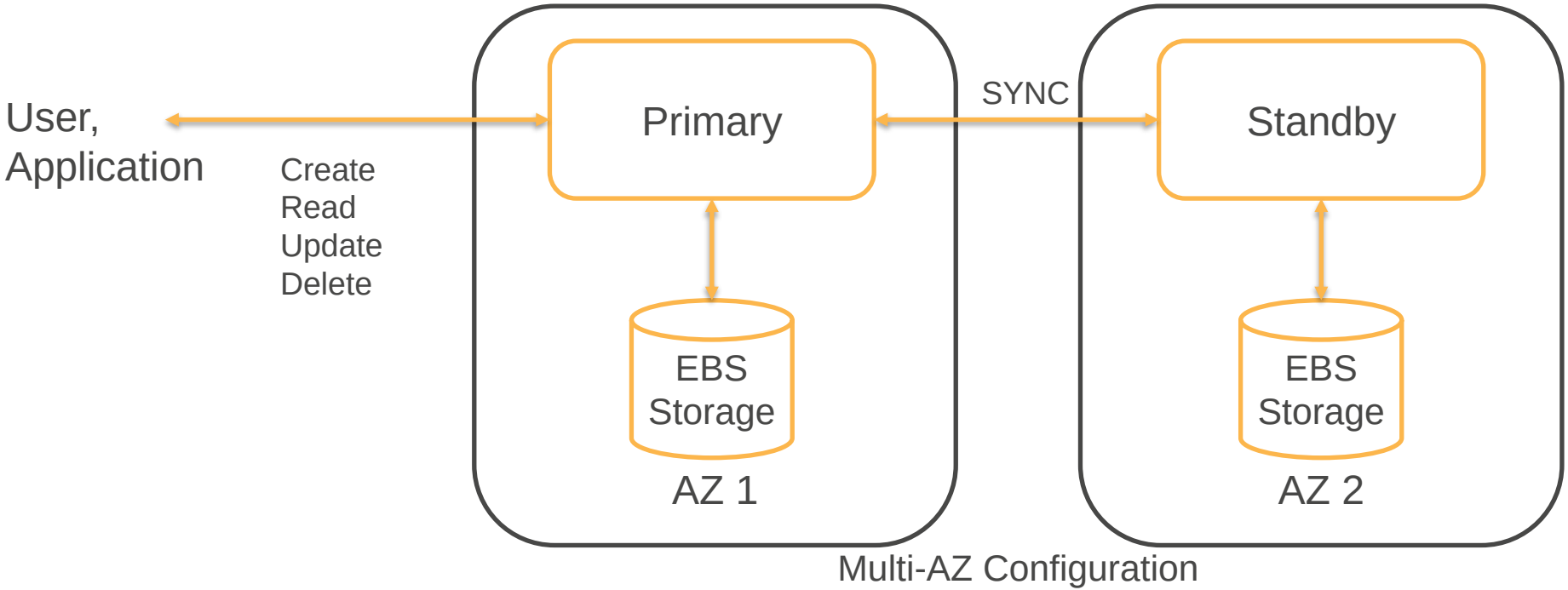
Automates time-consuming administrative tasks (hardware, installation, patching, backup)

Production ready database in minutes

Push button scaling (CPU, Memory, Storage)

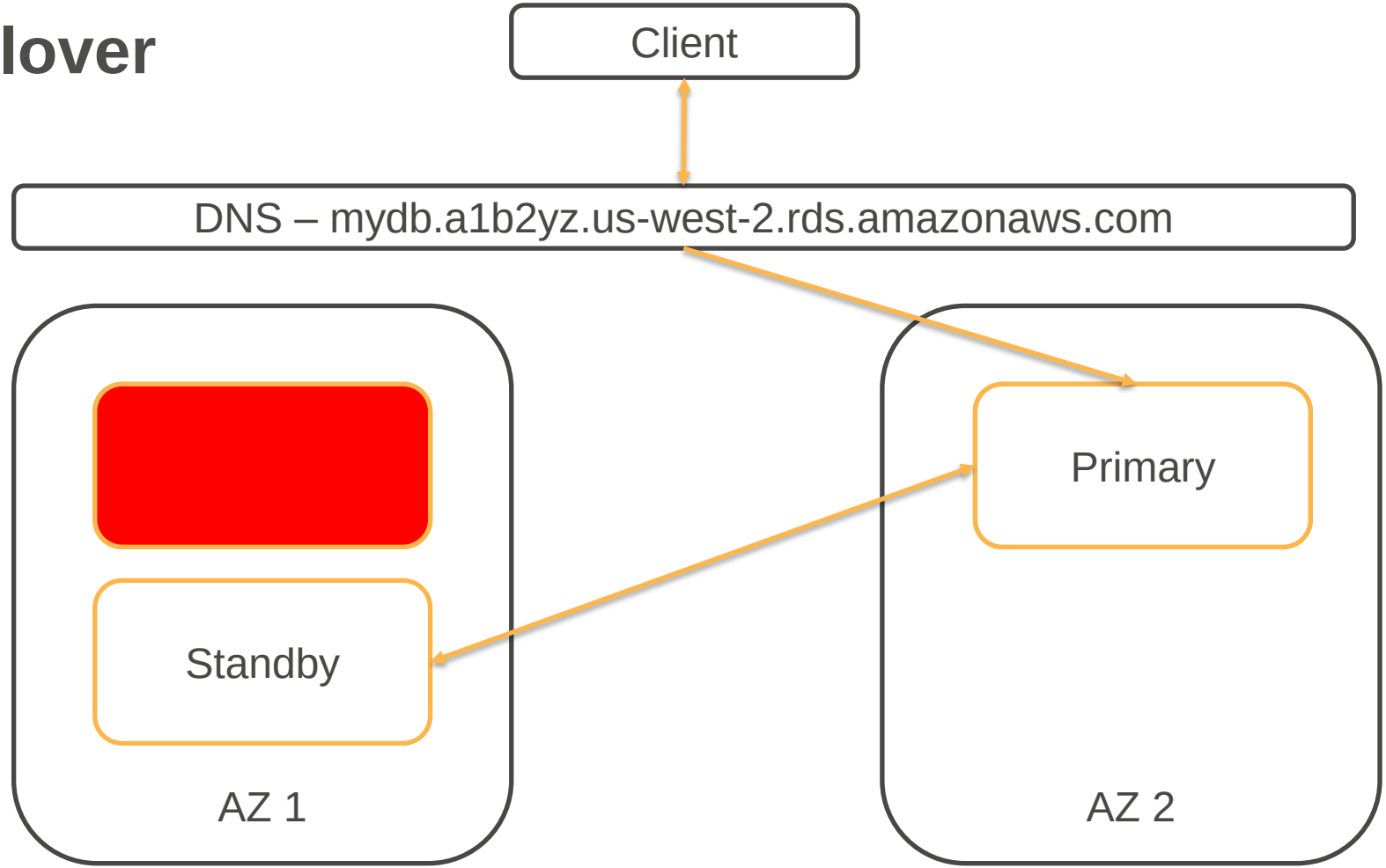
Six popular database engines: Aurora, MySQL, PostgreSQL, MariaDB, Oracle, SQL Server

Amazon Relational Database Service (RDS)

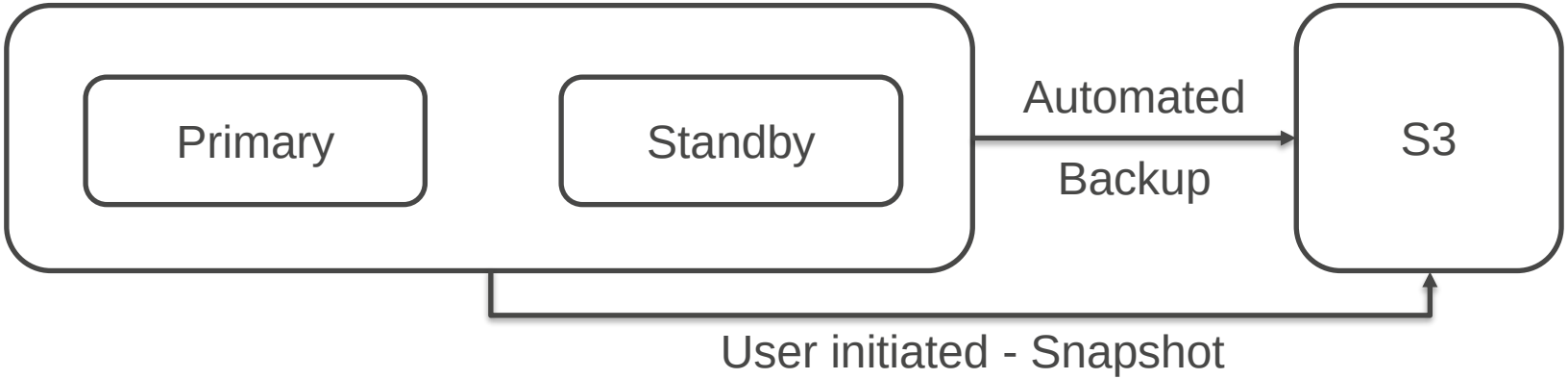


- Connect using DNS Name
- RDS maintains mapping between DNS Name and Primary Instance
- After failover, DNS is updated to point to new primary

Failover



RDS - Backup and Snapshot



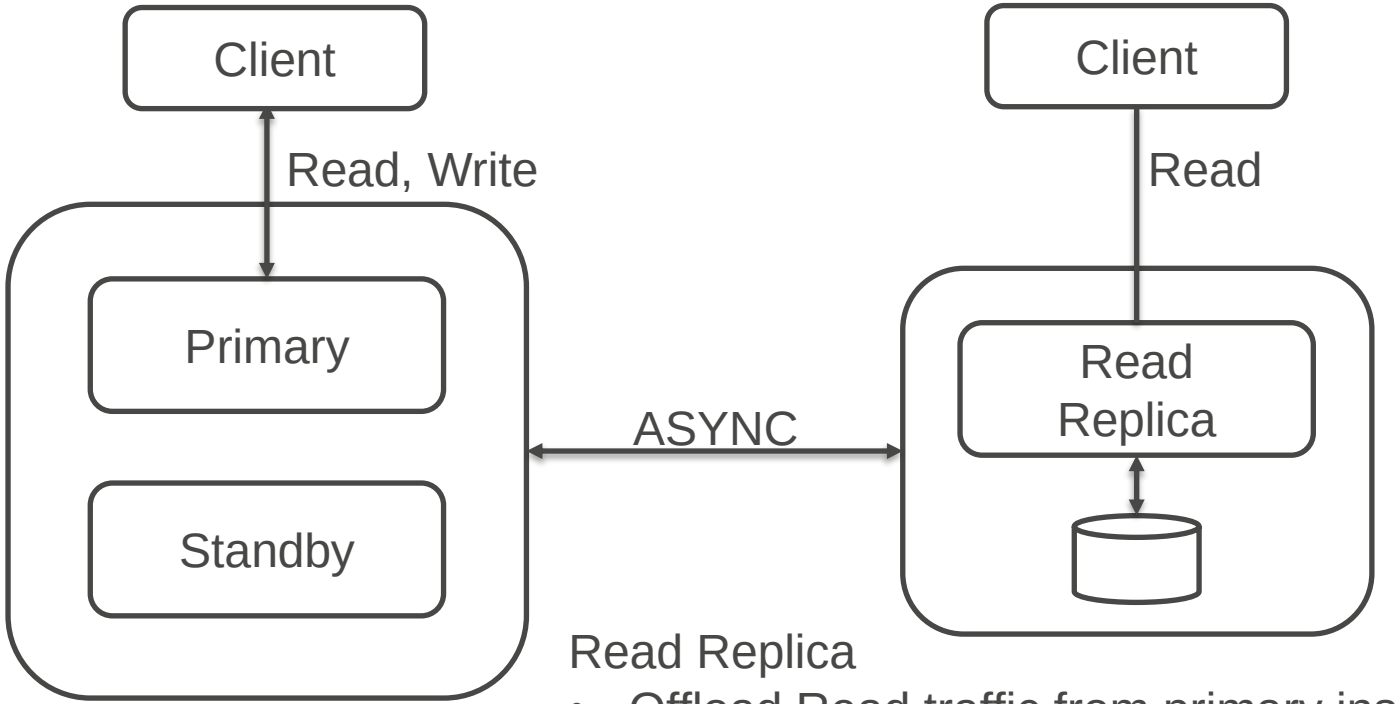
Automated Backup

- Configurable for a retention up to 35 days
- Last restorable time – typically within last 5 minutes
- Point-in-time restore up to specified second (to a new instance)

Snapshot

- User initiated
- Snapshot is kept until explicitly deleted
- Suitable for long term retention
- Copy to another region

RDS – Read Replica



Read Replica

- Offload Read traffic from primary instance
- Data can be stale
- One or more read replicas (depending on DB engine)

RDS Patching

"Amazon RDS will make sure that the relational database software powering your deployment stays up-to-date with the latest patches.“

You can specify a maintenance window that RDS can use for patching systems

RDS – Scaling CPU and Memory

- Specify desired CPU and Memory configuration and RDS takes care of scaling
- Completes in a few minutes (needs to spin up new instances)
- RDS performs failover during compute scaling (interruption to client for the duration of failover)

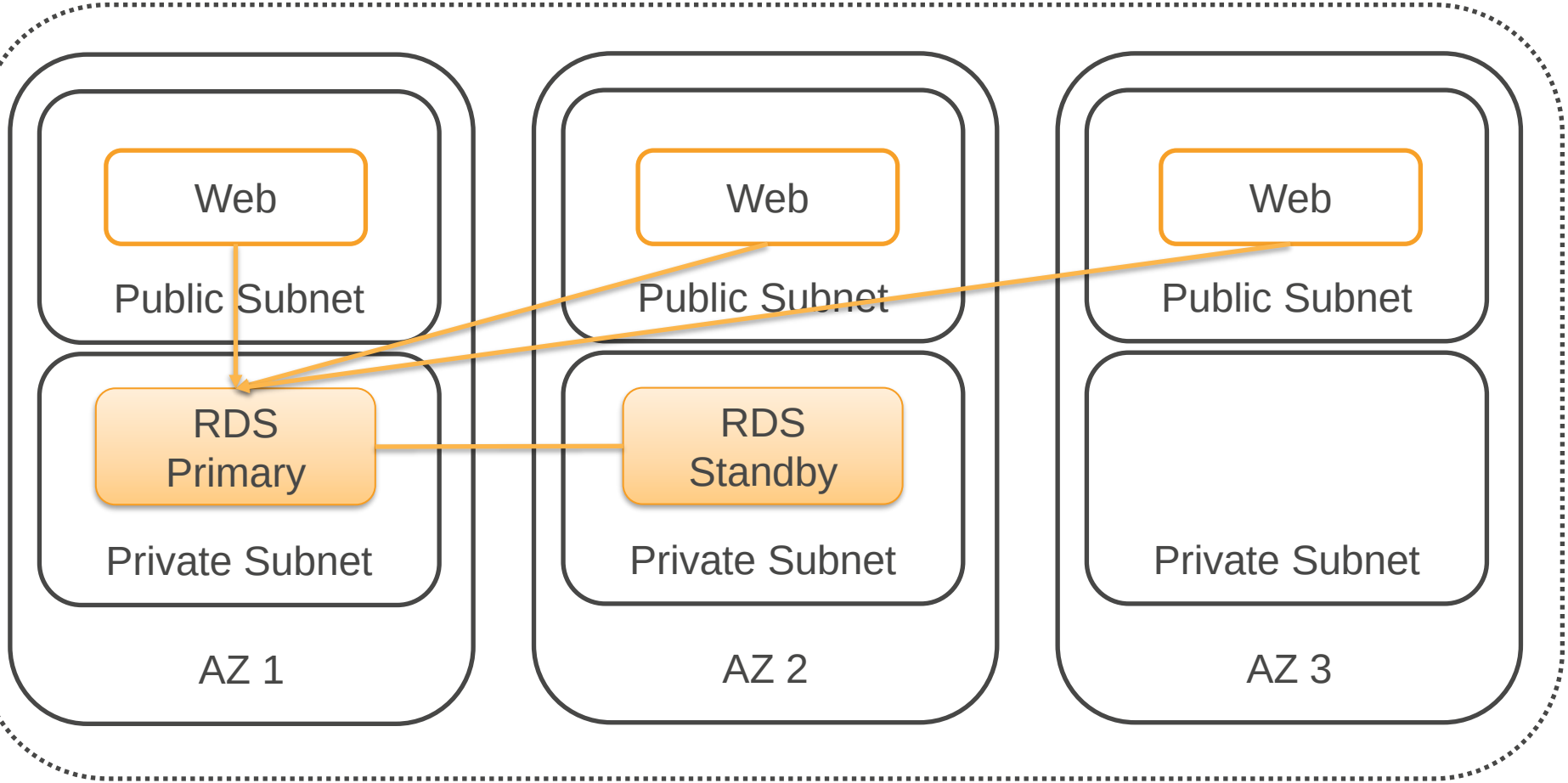
Scaling can be scheduled during next maintenance window or apply-immediate

RDS – Storage Scaling

- Storage can be scaled without interruption (zero-downtime)
- SQL Server up to 16 TB
- Aurora up to 64 TB
- MySQL, MariaDB, PostgreSQL, Oracle up to 32 TB

Scaling can be scheduled during next maintenance window or apply-immediate

RDS – Deployment



RDS – Network Security

- Deploy RDS in Private Subnet (unless your requirement is a publicly accessible RDS instance)
- Configure RDS Security Group to allow access from Web Server or Application Server Security Groups
- Assign a subnet in all Availability Zones to the DB Subnet Group
    - In case of extended AZ down or some other issue, RDS may choose to launch a replacement standby instance in a different AZ
- Connect from on-premises using Amazon DirectConnect or VPN

RDS – Permissions and Encryption

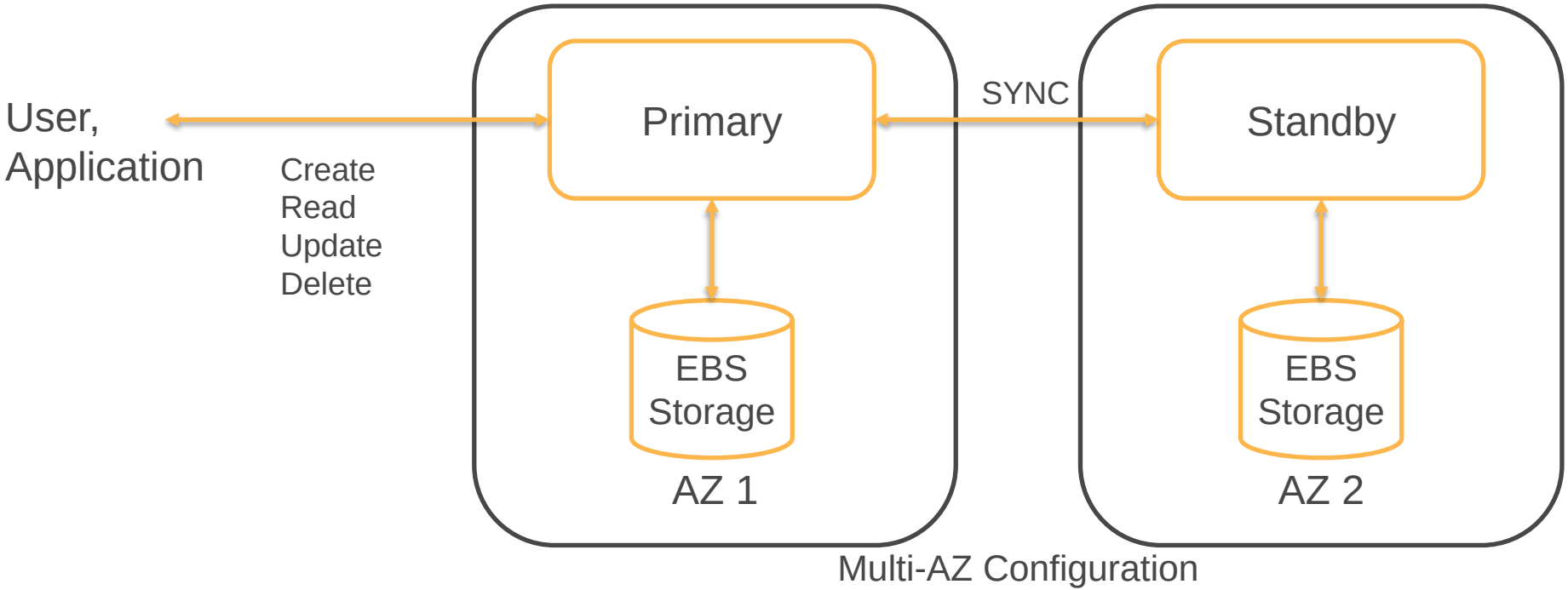
- IAM for Control Plane Access – who can create, manage, delete RDS database instances
- DB Specific User for Data Plane access – who can connect to the database, run SQL
- Optional encryption at rest using AWS Key Management Service (KMS)
- Optional encrypted connection support using SSL/TLS

RDS – Customization, Optimization

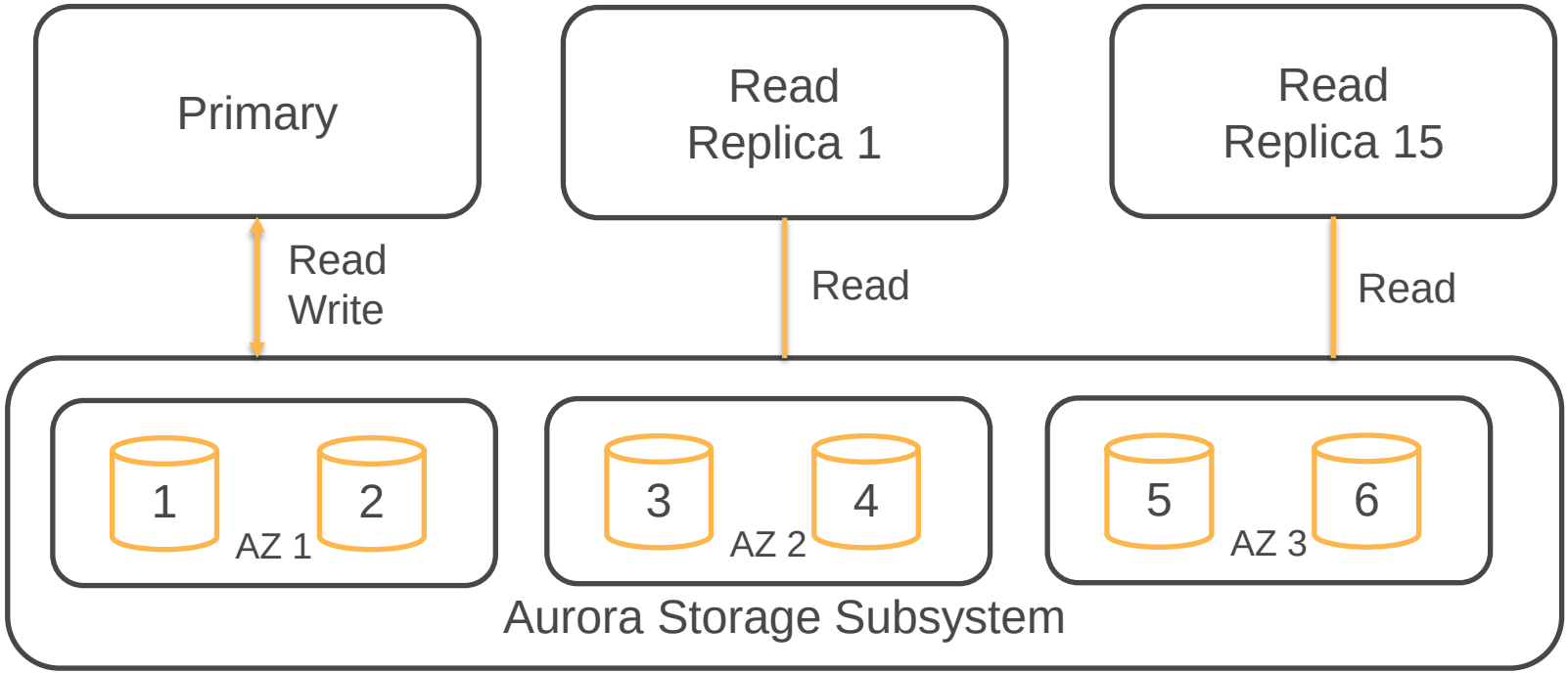
- You can customize RDS database instance and fine tune using *DB Parameter Groups*
- RDS provides best practice guidance by analyzing configuration and usage metrics
- Use Reserved Instances for long term use (1 to 3-year terms) at substantial discount
- To prevent configuration drifts, you can use AWS Config to record and audit changes to DB instance
- For monitoring, you can use CloudWatch

Amazon Aurora and Aurora Serverless

Traditional Relational Database Engine



Amazon Aurora



Aurora vs other Relational Databases

- Storage Subsystem that automatically maintains six copies of data across three availability zones
- Any changes made by Primary instance is replicated automatically
- Low latency Read Replica instances (lag time often in single digit millisecond)
- When the Primary fails,
    - A Read Replica is promoted as the new primary (typically under 60 seconds)
    - If Read Replica is not there, a new replacement primary is launched

Aurora Features

- MySQL and PostgreSQL Compatibility Modes
- Up to five times faster than standard MySQL database
- Up to three times faster than standard PostgreSQL database
- Security, Availability, Reliability of commercial databases at 1/10th cost
- Support for up to 15 low latency read replicas
- Global Database - Multi-Region Replication (fast local access, disaster recovery) for globally distributed applications

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

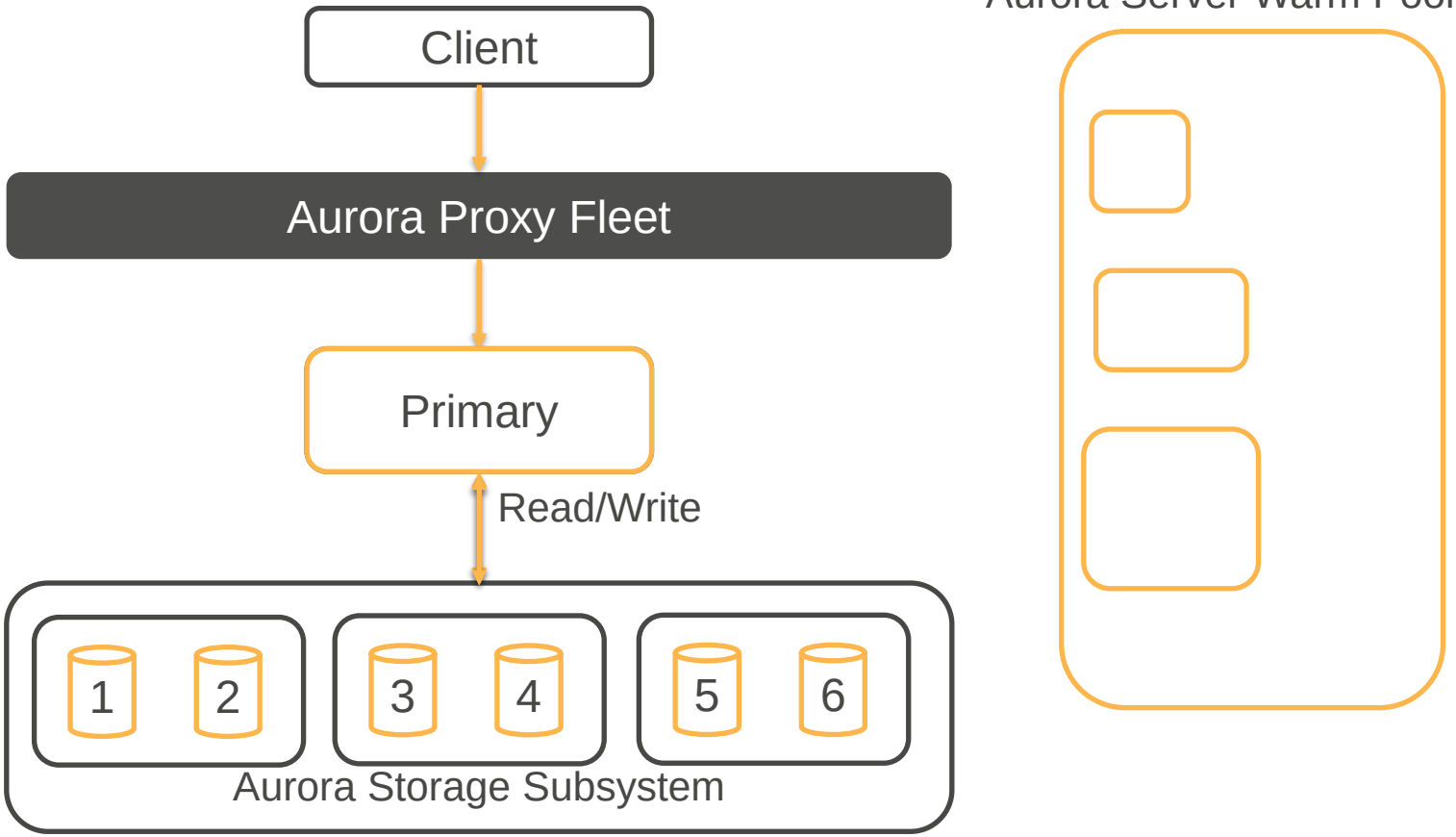
- Points to Read Replicas
- Suitable for Reads
- Multiple Read Replicas are load balanced at connection level

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

Instance Endpoint

- Points to Individual Aurora Instance

Amazon Aurora Serverless



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

Aurora Serverless

- 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

NoSQL Databases

DynamoDB, Cassandra, DocumentDB

DynamoDB

- Key-value NoSQL datastore
- Flexible schema - only primary key needs to be defined – all columns/attributes are flexible
- Consistent performance at any scale – single digit millisecond

Example: Movie Data

```
{
"year": 2013,
"title": "Rush",
"info": {
    "directors": ["Ron Howard"],
    "release_date": "2013-09-02T00:00:00Z",
    "rating": 8.3,
    "genres": ["Action","Biography",
               "Drama","Sport"],
    "actors": ["Daniel Bruhl","Chris Hemsworth",
               "Olivia Wilde"]
    }
}
```

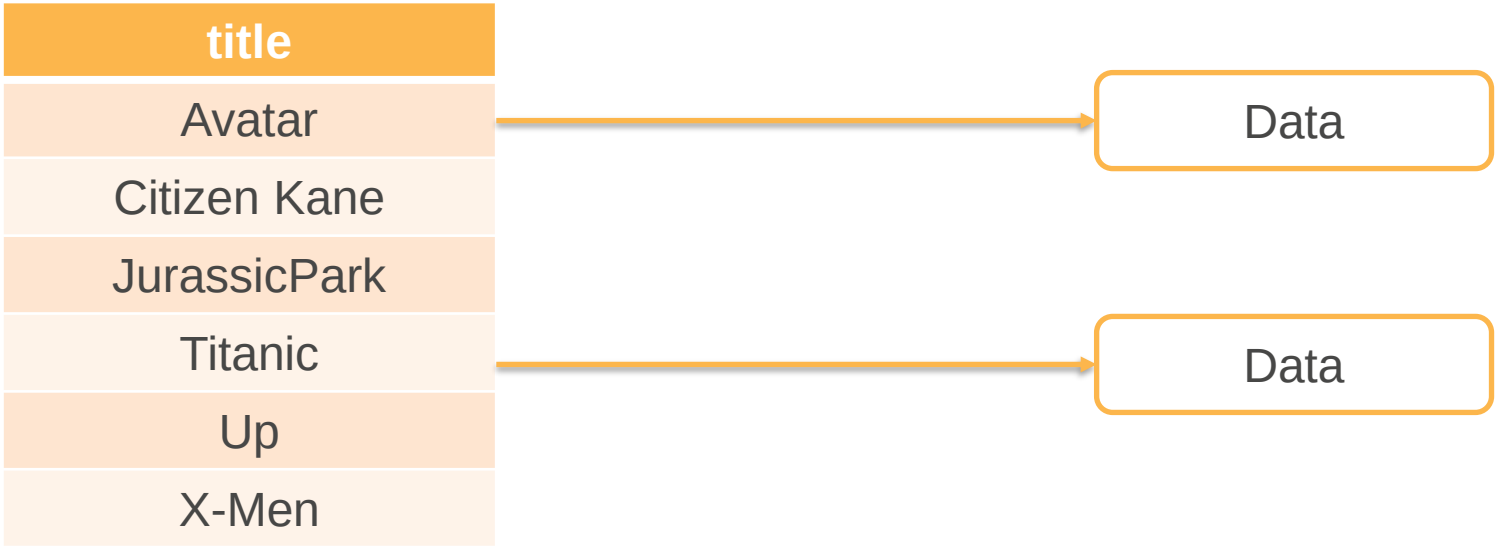
Data Sample:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.html

Primary Key

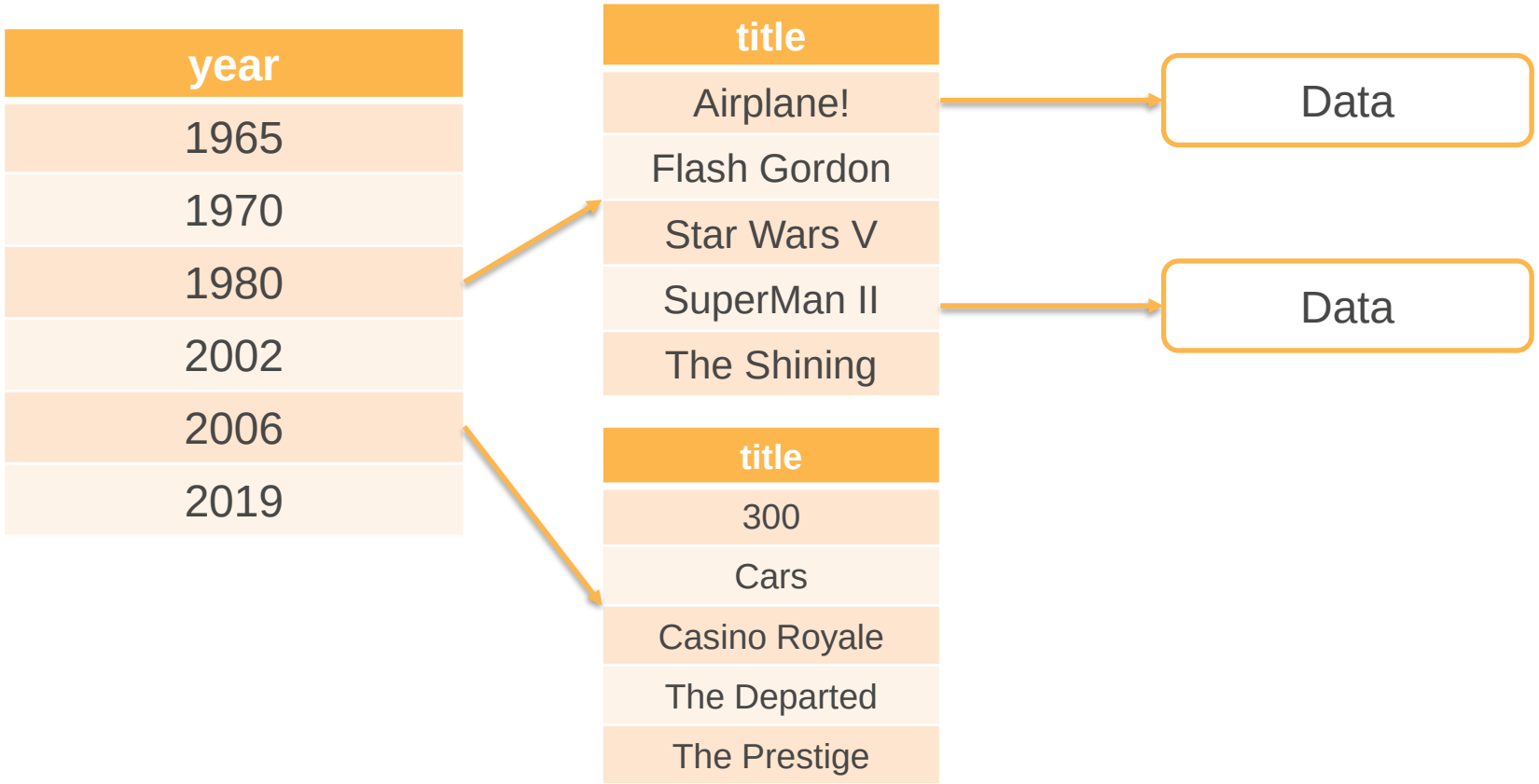
Simple – Single attribute

Composite – Two attributes (partition key, sort key)

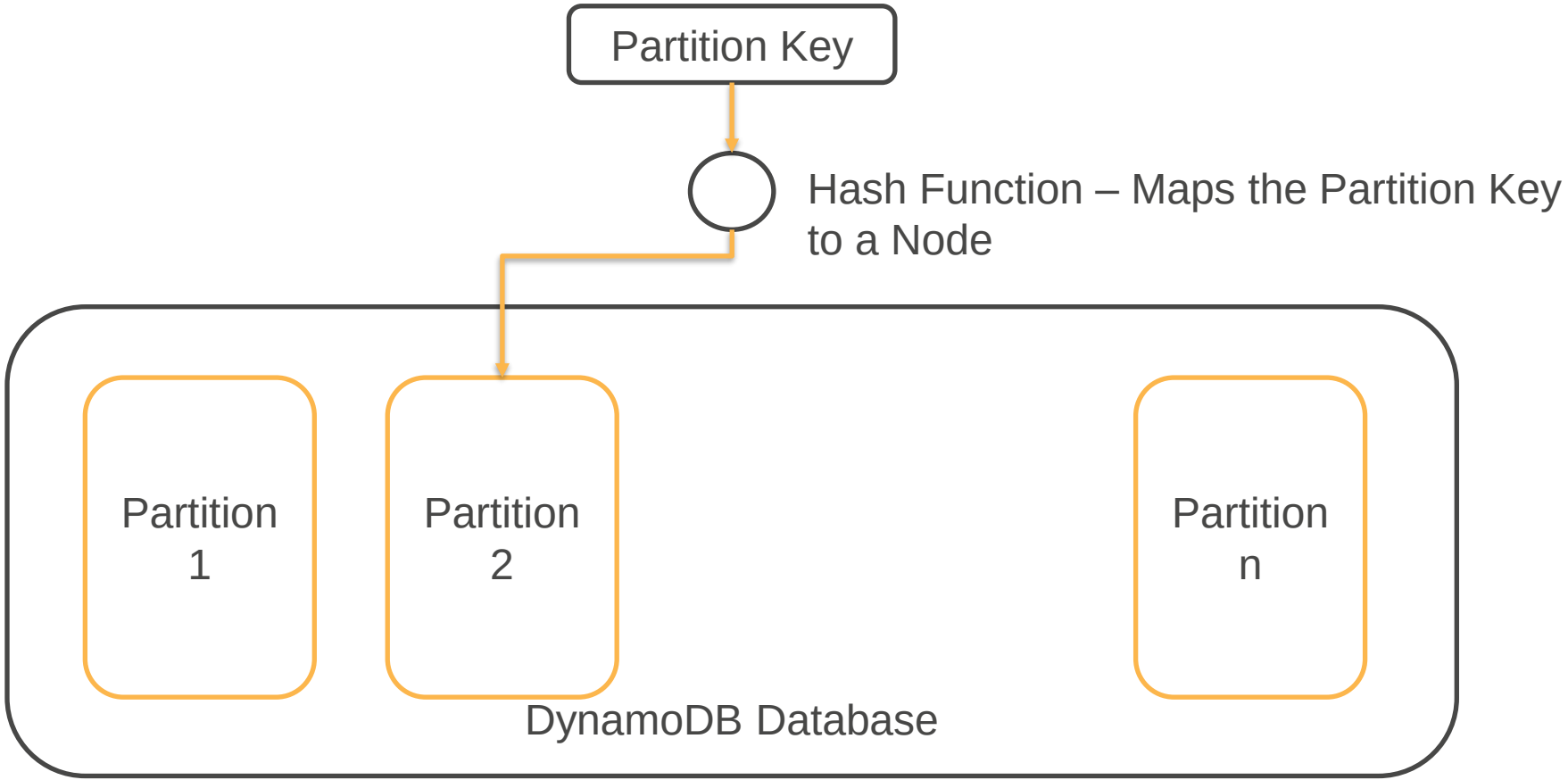
Simple Primary Key - title



Composite Primary Key – year, title



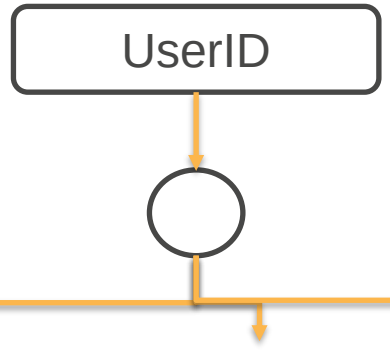
Scale-Out Processing



Game Score Table

UserID	GameTitle	Country	Other attributes
101	G1	USA	
101	G2	USA	
102	G1	USA	
102	G2	USA	
103	G1	USA	
103	G2	USA	
104	G1	USA	
104	G2	USA	

Game Score Table – UserID, GameTitle



Partition 1

UserID	Title
101	G1
101	G2
104	G1
104	G2

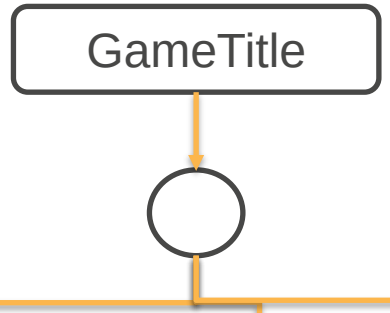
Partition 2

UserID	Title
102	G1
102	G2

Partition 3

UserID	Title
103	G1
103	G2

Game Score Table – GameTitle, UserID



Partition 1

Title	UserID
G1	101
G1	102
G1	103
G1	104

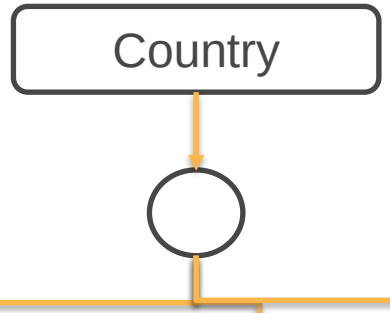
Partition 2

Title	UserID
G2	101
G2	102
G2	103
G2	104

Partition 3

Title	UserID

Game Score Table – Country, UserID



Partition 1

Country	UserID
USA	101
USA	102
USA	103
USA	104

Partition 2

Country	UserID

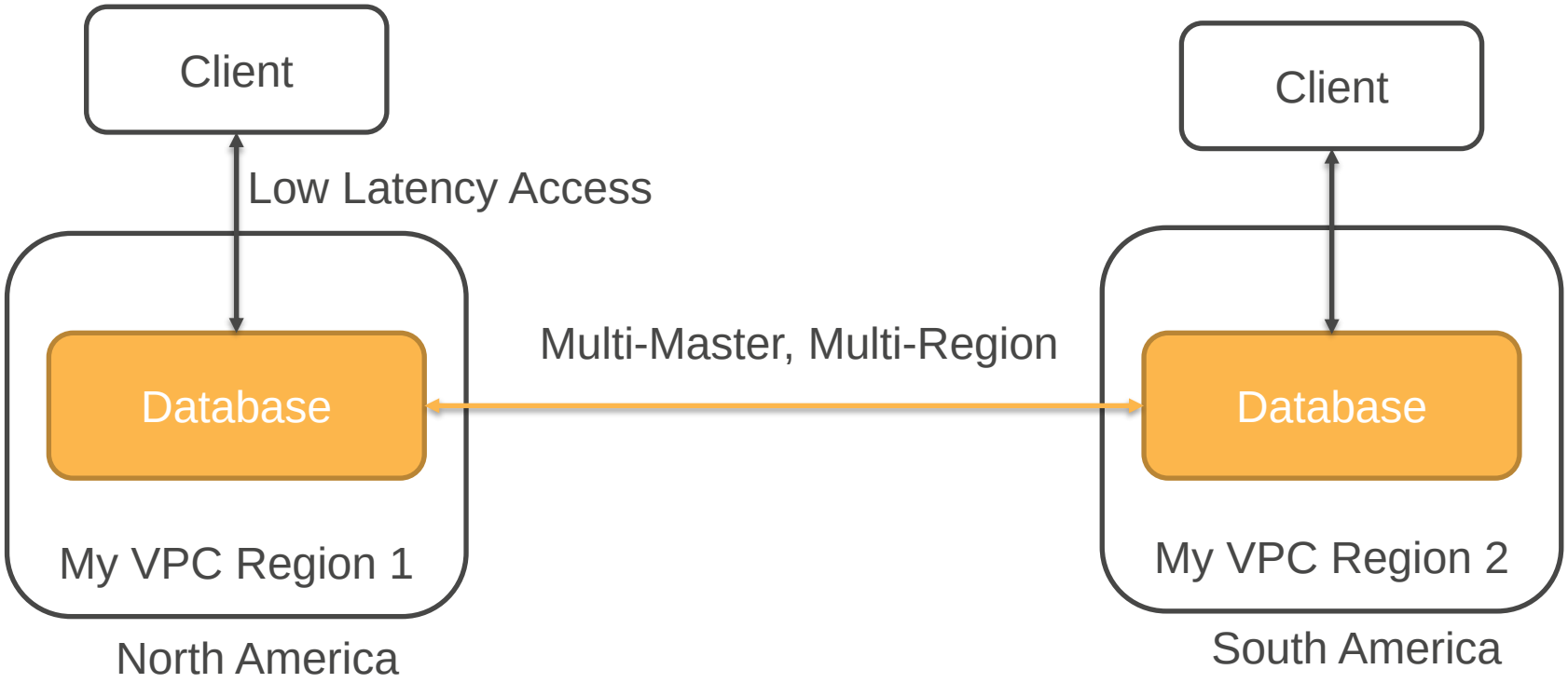
Partition 3

Country	UserID

DynamoDB Features

- Automatic replication of data across multiple-availability zones in a region
- Global Tables – multi-master, multi-region replication -Fast local access across different regions
- ACID Transaction Support
- Point-in-Time Recovery – Automated Continuous Backup (35 days retention)
- On-Demand Backup/Snapshot for long term retention
- Automatic deletion of expired items – Time To Live
- Limits - Item size cannot exceed 400 KB

DynamoDB Global Table - Multi-Region, Multi-Master



Transactions

DynamoDB supports ACID Transactions - Atomicity, Consistency, Isolation, Durability

Transactions are useful when you want to insert, delete or update multiple items as a single logical operation

"DynamoDB provides native, server-side support for transactions, simplifying the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables"

https://aws.amazon.com/dynamodb/features/

Create, Read, Update, Delete

- GetItem, BatchGetItem
- Query
- Scan
- PutItem, UpdateItem, DeleteItem
- BatchWriteItem

Reading Data from DynamoDB

GetItem

- Read one item using Primary Key
- Fastest Access
- Direct access to Physical location of the item

BatchGetItem

- Batch read up to 100 items (using Primary Keys)
- Reduce round-trip between Client and DynamoDB
- Parallel Read, Retrieve Items from multiple tables (not joins)

Example: Retrieve (1980, Flash Gordon) details from movies table – primary key is year and title

Reading Data from DynamoDB

Query

- Used when you define a composite primary key for your table
- Query by Partition Key – returns all items for a partition key
- Optional - specify condition for Sort Key (Equal, Prefix, Range, greater than, less than and so forth)
- Efficient Access to the Partition and retrieve items that match Sort Key Condition

Example: Retrieve all movies released in (1980) from movies table – primary key is year and title

Reading Data from DynamoDB

Scan

- Retrieves all items in the specified table
- Very expensive and in-efficient
- Can consume large amount of system resources

Example: Retrieve all movies with ratings > 7.0 from movies table.
Primary key is year and title.
No index defined for ratings in the movies table

DynamoDB – Create, Update, Delete Data

PutItem – Create an Item

UpdateItem – Update an Item

DeleteItem – Delete an Item

BatchWriteItem

- Batch up to 25 individual PutItem and DeleteItem requests
- Put or delete items in multiple tables
- Parallel operation

All these operations require entire primary key

DynamoDB – Condition Writes, Atomic Counters

Conditional writes – specify a condition that needs to be met before an item is updated

Comparable to WHERE clause in SQL

Atomic Counters – are useful for incrementing and decrementing numeric attributes

Secondary Index

- Efficiently query by non-primary key attributes of the base table
- Secondary Index requires a partition key and sort key
- Specify other attributes that need to projected (copied) from base table

  Base Table Primary Key is always projected
- Index is updated when Add/Update/Delete operations are performed in the base table
- Secondary index is comparable to any other DynamoDB table – you can run Query and Scan operations on the index

  DynamoDB does NOT use Index automatically (Relational Database systems automatically use appropriate index when you query the base table)

Global Secondary Index

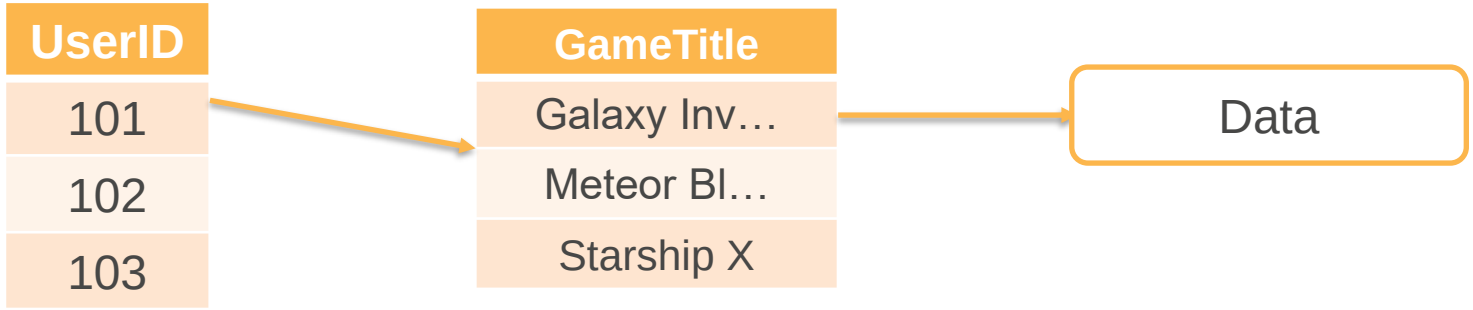
1. Index Partition Key is different from base table partition key
2. Global Secondary Index can be added to existing tables

For example,

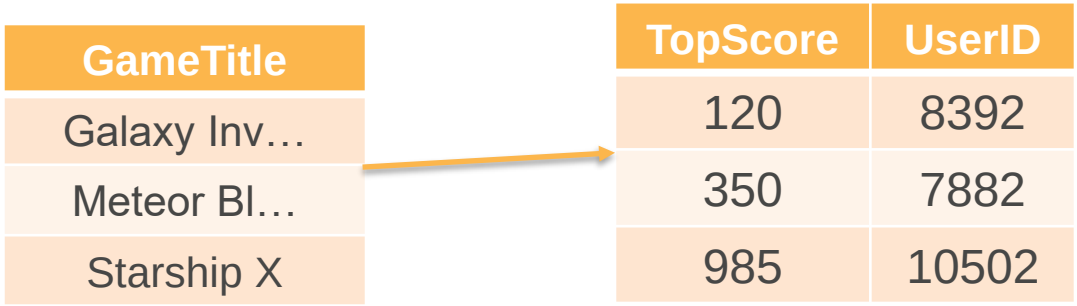
GameScores Table - Primary key UserID (partition key), GameTitle (sort key)

GameTitle Index – GameTitle (partition key), TopScore (sort key) – Easily query top scoring users for each game

Game Score Table – Primary Key (UserID, GameTitle)



GameTitle Global Index (GameTitle, TopScore)



- Base Table and Index have the different Partition Key attribute

Local Secondary Index

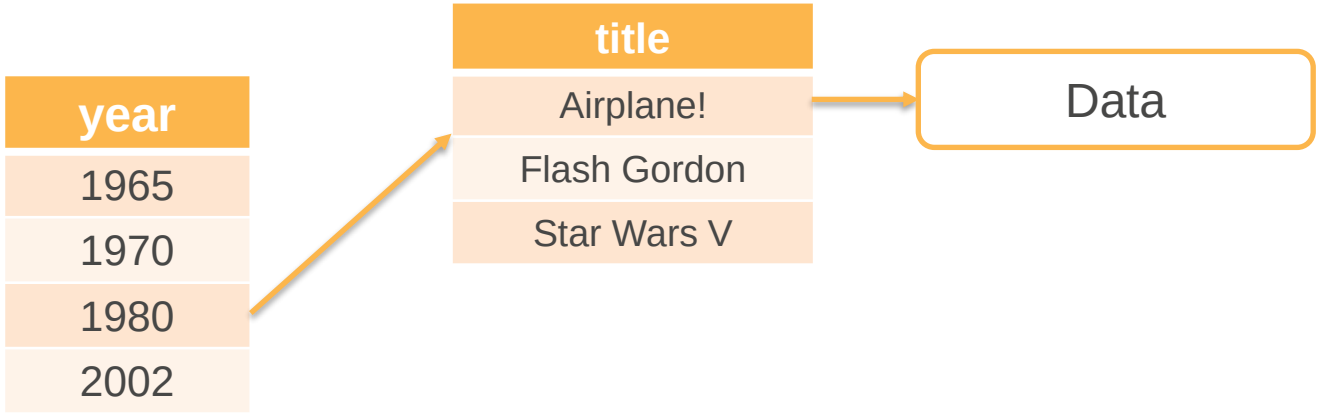
1. Index Partition Key is same as base table partition key
2. Local Secondary Index can be added only at table creation time

For example,

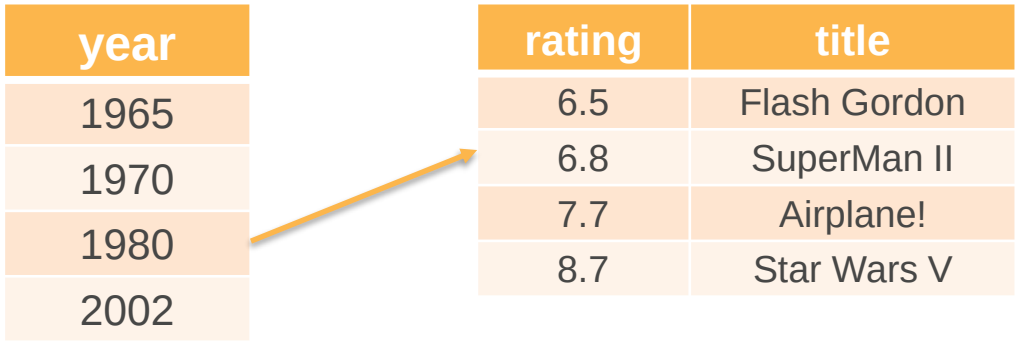
Movie Table - primary key: year (partition key), title (sort key)

YearRating Index - year (partition key), rating (sort key) – easily query top rated movies for each year

Movie Table Key – Primary Key (year, title)



YearRating Local Index (year, rating)



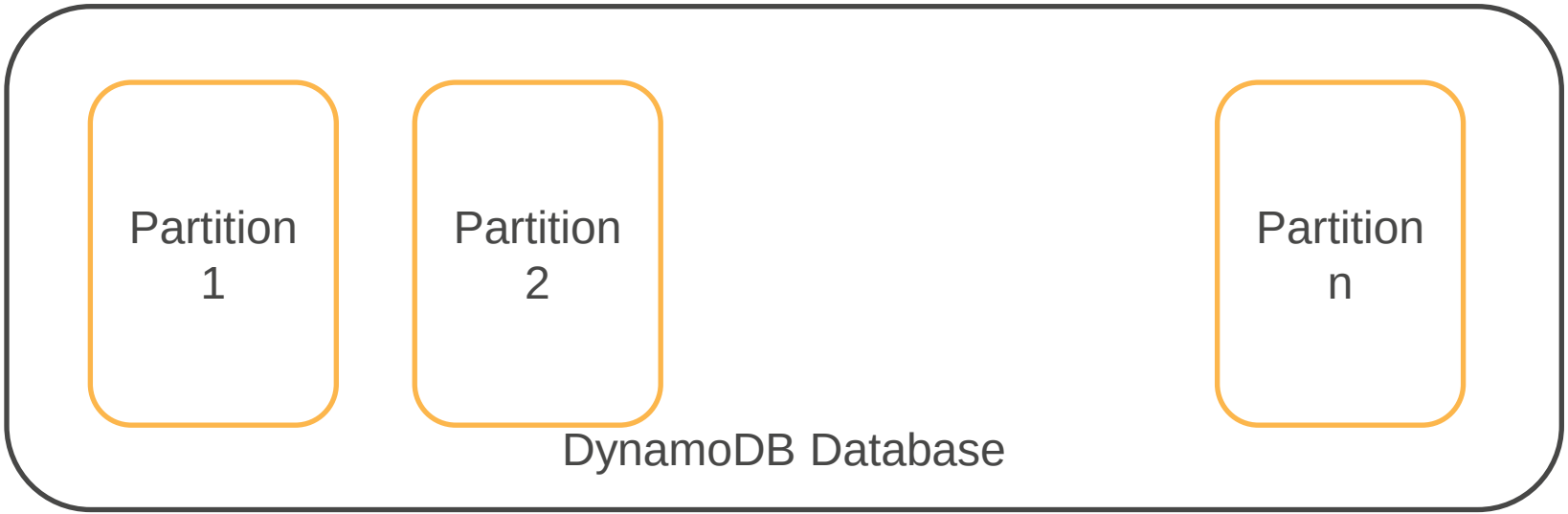
- Base Table and Index have the same Partition Key attribute

DynamoDB

- Consistency Model – Table, Index
- Capacity Management – On-demand, Provisioned
- Streams
- DAX – In-memory acceleration
- Best Practices

Consistency with Scale-Out Processing

- Data is stored in partitions
- Partitions are replicated across multiple-AZs
- DynamoDB request can go to any of the nodes
- Changes may take up to a second to reach consistency across all nodes



Read Consistency – Base Table

Eventual Consistency

- Default
- May not return recently completed write
- Maximize read throughput, Less expensive

Strong Consistency

- Returns up-to-date data
- Twice as expensive as Eventual Consistency

Specify required consistency - Get, Batch Get, Query, Scan

Read Consistency – Index

Global Index Query, Scan

- Eventually Consistent

Local Index Query, Scan

- Default Eventual Consistency
- Supports Strong Consistency

DynamoDB Capacity Management

DynamoDB supports two types of capacity management

- On-demand
- Provisioned Capacity

On-Demand Capacity Management

- For less predictable workloads, on-demand capacity mode automatically takes care of managing capacity

  Example: sales promotion, new product launch

- Pay only for what you consume

"instantly accommodates your workloads as they ramp up or down to any previously reached traffic level."

"traffic level hits a new peak, DynamoDB adapts rapidly to accommodate the workload."

Provisioned Capacity

- For predictable workload, you can set the required read and write capacity
- Reserve capacity for discounts – need to commit to a time period
- Optionally, configure AutoScaling to adjust capacity between specified lower and upper limits and at target utilization level
- Manage Cost for predictable workload

Capacity Calculation

Read Capacity Unit (RCU)

- One RCU = One strongly consistent read per second or two eventually consistent reads per second
- Read Size = 4 KB
- Transactional Reads consume twice as much capacity

For example, if item size is 8 KB

Strongly consistent read would require 2 RCUs

Eventual consistent read would require 1 RCU

Capacity Calculation

Write Capacity Unit (WCU)

- One WCU = One write per second
- Write Size = 1 KB
- Transactional Write consume twice as much capacity

For example, if item size is 8 KB

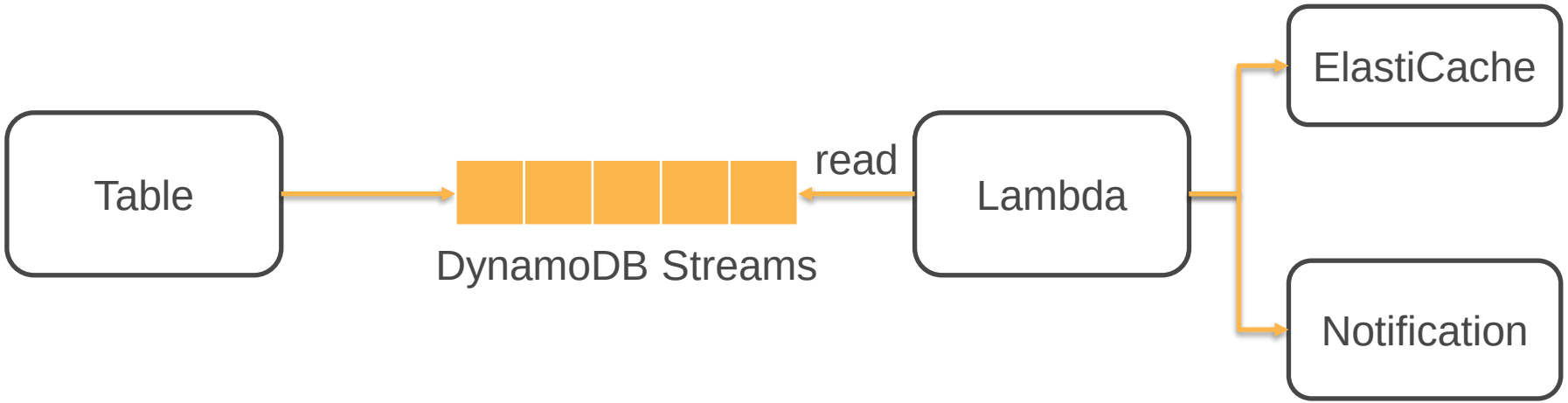
Writing the item would consume 8 WCUs

If it is part of a transaction, it would require 16 WCUs

DynamoDB Streams

“DynamoDB Streams captures time-ordered sequence of item-level modifications in any DynamoDB table and store this information in a log for up to 24 hours”

https://aws.amazon.com/dynamodb/features/

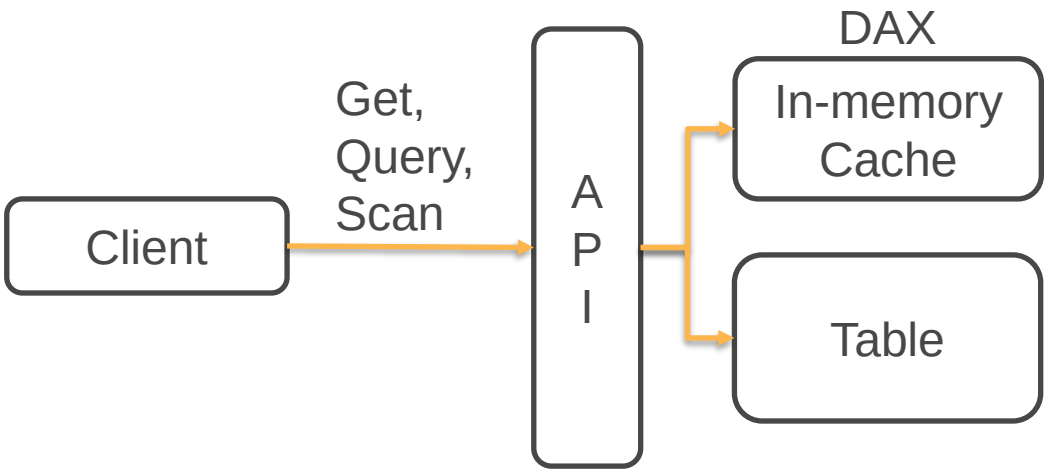


DynamoDB Accelerator (DAX)

In-memory acceleration for eventually consistent reads

API compatible – automatically served from cache

Increase throughput for read-heavy workloads



DynamoDB Use Cases

- User profile store
- Game states
- Leaderboards
- Retail Shopping Cart
- Inventory tracking and fulfilment
- User Transactions

DynamoDB Best Practices

- Fast performance – only when accessing by primary key
- Most applications require only one base table
- Keep secondary indexes to a minimum
- For Partition key, pick an attribute that contains large number of unique values – ensures all partitions are equally used and prevents “hot partitions”
- For Time-Series data, create a table for each time period – assign appropriate capacity and reduce capacity for older tables

Cassandra, DocumentDB

Amazon Managed Cassandra

AWS managed open source Apache Cassandra

Move Cassandra workloads to AWS Cloud

Performance Benefits are comparable to DynamoDB

AWS recommends Cassandra for: industrial equipment data collection, and other use cases that require high performance and large number of columns

Cassandra versus DynamoDB

- DynamoDB primary key is made up of single attribute partition key and optional single attribute sort key. Cassandra supports multi-column partition and sort keys
- DynamoDB max item size is 400KB – Cassandra has a theoretical limit of 2GB per column. However, general practice is not to exceed few MBs.
- Cassandra also supports large number of columns – DynamoDB even though supports large number of attributes, it is constrained by 400KB size limit per item

Amazon DocumentDB

“Amazon DocumentDB (with MongoDB compatibility) is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads.”

DocumentDB emulates MongoDB API and it is not true port of open source code. Currently, there is a drift in the direction of MongoDB and DocumentDB.

ElastiCache

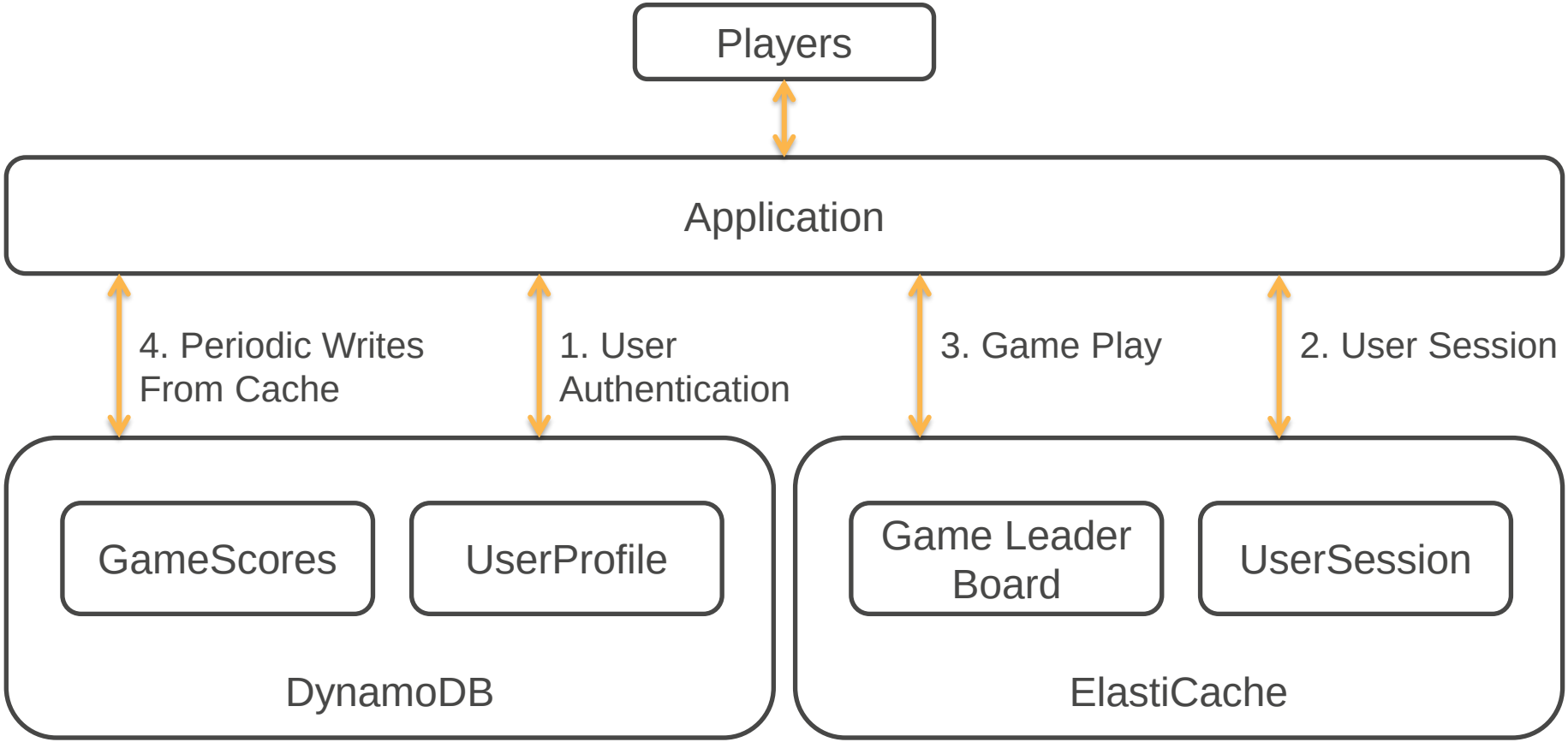
In-memory data store

Amazon ElastiCache

- In-memory datastore with sub-millisecond latency
- Ideal for frequently read data, reduce read-traffic going to database, buffer high-frequency writes and periodically reconcile with backend database

  Uses: Product reviews and rating, Caching, Session Management, Gaming leaderboards, geospatial applications
- Deploy in your VPC – Network isolation and security
- Choice of engines: Memcached, Redis

Game Leader Boards (READs/WRITEs)



MemCached Features

- Key-value store
- Scales up to 20 nodes and 12.7 TB
- Sub milli-second latency

Redis Features

- In-memory datastore with advanced data structures: Strings, Lists, Sorted Sets, Hash, Bit Arrays

  Sorted Sets can be used to easily Game Leader Boards – keep a list of players sorted by rank.
- Built-in commands for Geospatial data
  - Distance between two places or persons
  - Find all places within a given distance from a point
- Sub milli-second latency
- Scales up to 250 nodes and 170 TB

Redis High Availability Features

- Pub-Sub and Messaging

  For example, High performance chat rooms, server to server communication, social media feeds
- Read Replica across multiple Availability Zones
- Detects primary node failure and automatically promotes replica as primary
- Backup, Restore
- Export to another region
- Lua scripting support

Amazon Redshift

Data Warehouse - Redshift

- Peta Byte Scale Massively Parallel Relational Database
- Cluster consists of Leader Node and Multiple Compute Nodes

  Available Storage = Storage per Compute Node X Number of Compute nodes
- Columnar Storage
- Targeted Data Compression
- Powerful SQL based Analytics
- With Redshift Spectrum - query can span tables in Redshift and files stored in S3 Data Lake

Chandra Lingam

57,000+ Students

aws certified Solutions Architect Associate

aws certified Developer Associate

aws certified SysOps Administrator Associate

aws certified Solutions Architect Professional

aws certified Machine Learning Specialty

For AWS self-paced video courses, visit:

https://www.cloudwavetraining.com/

Cloud Wave LLC