CLOUD COMPUTING APPLICATIONS

# Cloud Databases – Managed RDBMS

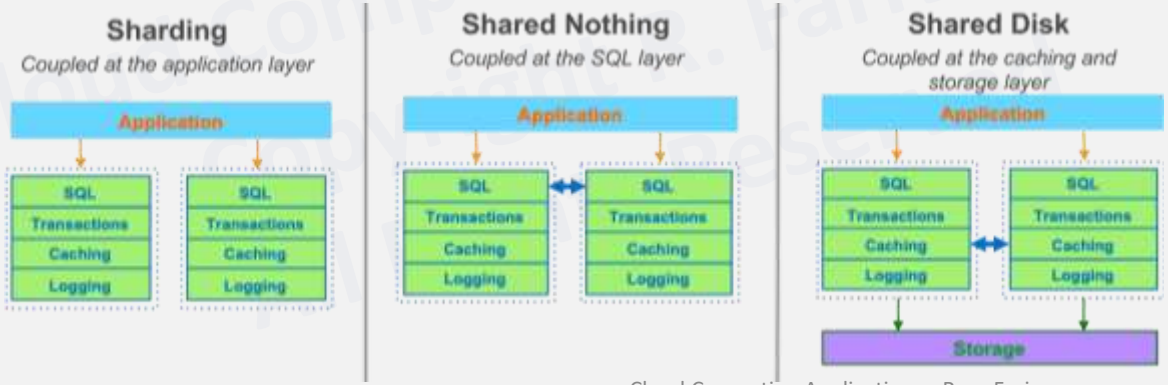
Prof. Reza Farivar

# Relational Cloud Databases

- For decades, managing a relational database has been a high-skill, labor-intensive task
- Relational databases store data with predefined schemas and relationships between them
- These databases are designed to support ACID transactions, maintain referential integrity and strong data consistency.
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- OLTP workloads
  - On- Line Transactional Processing (OLTP)

# Relational Databases

- In large systems, failures are a norm, not an exception
- Users want to start with a small footprint and then grow massively without infrastructure limiting their velocity
- Replication
  - Storage (SAN, NAS, Aurora)
  - Database
  - Application



# Sharding

- Sharding: Split data set by certain criteria and store such “shards” on separate “clusters”
  - Sharding can be considered an embodiment of the "share-nothing" architecture and essentially involves breaking a large database into several smaller databases
- One common way to split a database is splitting tables that are not joined in the same query onto different hosts
- Another method is duplicating a table across multiple hosts and then using a hashing algorithm to determine which host receives a given update

# Managed Relational Databases

- Traditional single server databases running on a virtual machine
  - AWS RDS: MySQL, PostgreSQL, MariaDB, Oracle, MS SQL server
  - Azure SQL Database, Database for MySQL, PostgreSQL, MariaDB
  - Google Cloud SQL
  - IBM Cloud Databases for PostgreSQL, DB2 on cloud
- Instances are fully managed, relational MySQL, PostgreSQL, and SQL Server databases
- Cloud provider handles replication, patch management, and database management to ensure availability and performance
- Availability through failover
- Horizontal Scalability through read replicas
  - Vertical scalability by using larger machines (64 processors, 400GB RAM)

# Managed Relational Databases

- Typically the database instance is accessible by most compute resources in the cloud provider's network
  - Virtual Machines (AWS EC2, Azure VMs, Google Compute Engine)
  - PaaS (Aws Elastic beanstalk, Google App Engine
  - Serverless (AWS lambda, Google Cloud Functions, Azure functions, etc.)
- Over the internet
  - SQL Proxy
  - Google Cloud SQL Proxy for public interfacing
    - ./cloud_sql_proxy -instances=***INSTANCE_CONNECTION_NAME***=tcp:3306 &
    - import pymysql
      connection = pymysql.connect(host='127.0.0.1',
      user='***DATABASE_USER***',
      password='***PASSWORD***',
      db='***DATABASE_NAME***')
- Encryption at rest and in transport