AWS ElastiCache

What is Cache?

- Caching is a technique to store frequently accessed information in a temporary memory location on a server. Read-intensive web applications are the best use-case candidates for a cache service.
- There are a number of caching servers used across applications, the most notable are Memcached, Redis, and Varnish

Caching

There are various ways to implement caching using those technologies.

- However, with such large number of organizations moving their infrastructure to cloud, many cloud vendors are also providing caching as a service.

- Amazon ElastiCache is one such popular web caching service which provides users with Memcached or Redis-based caching that supports installation, configuration, HA, Caching failover and clustering.

What is ElastiCache ?

- ElastiCache is a web service that makes it easy to deploy, Operate and scale an in-memory cache in the cloud.

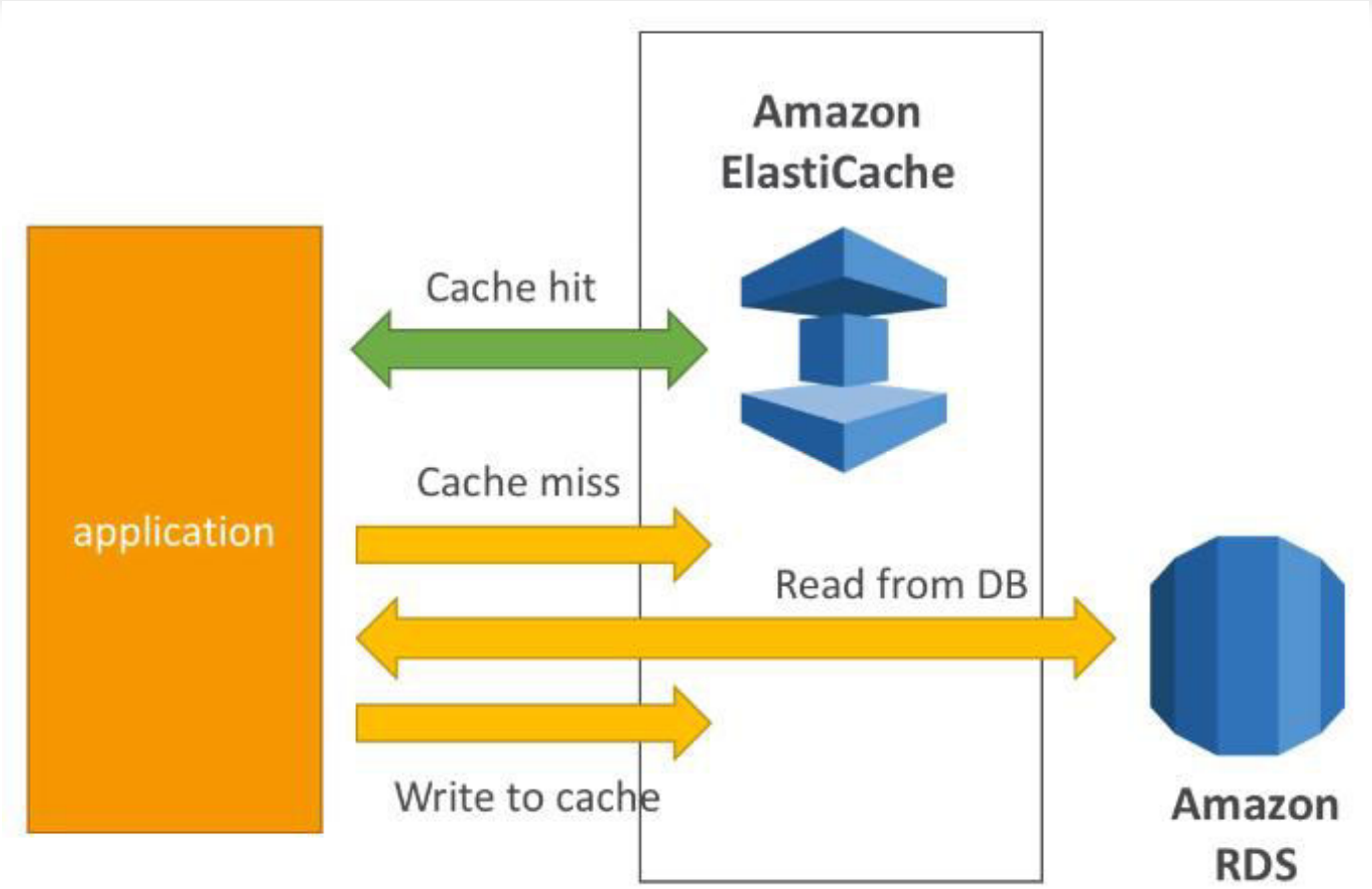
- The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based database.

- ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads

What is ElastiCache ?

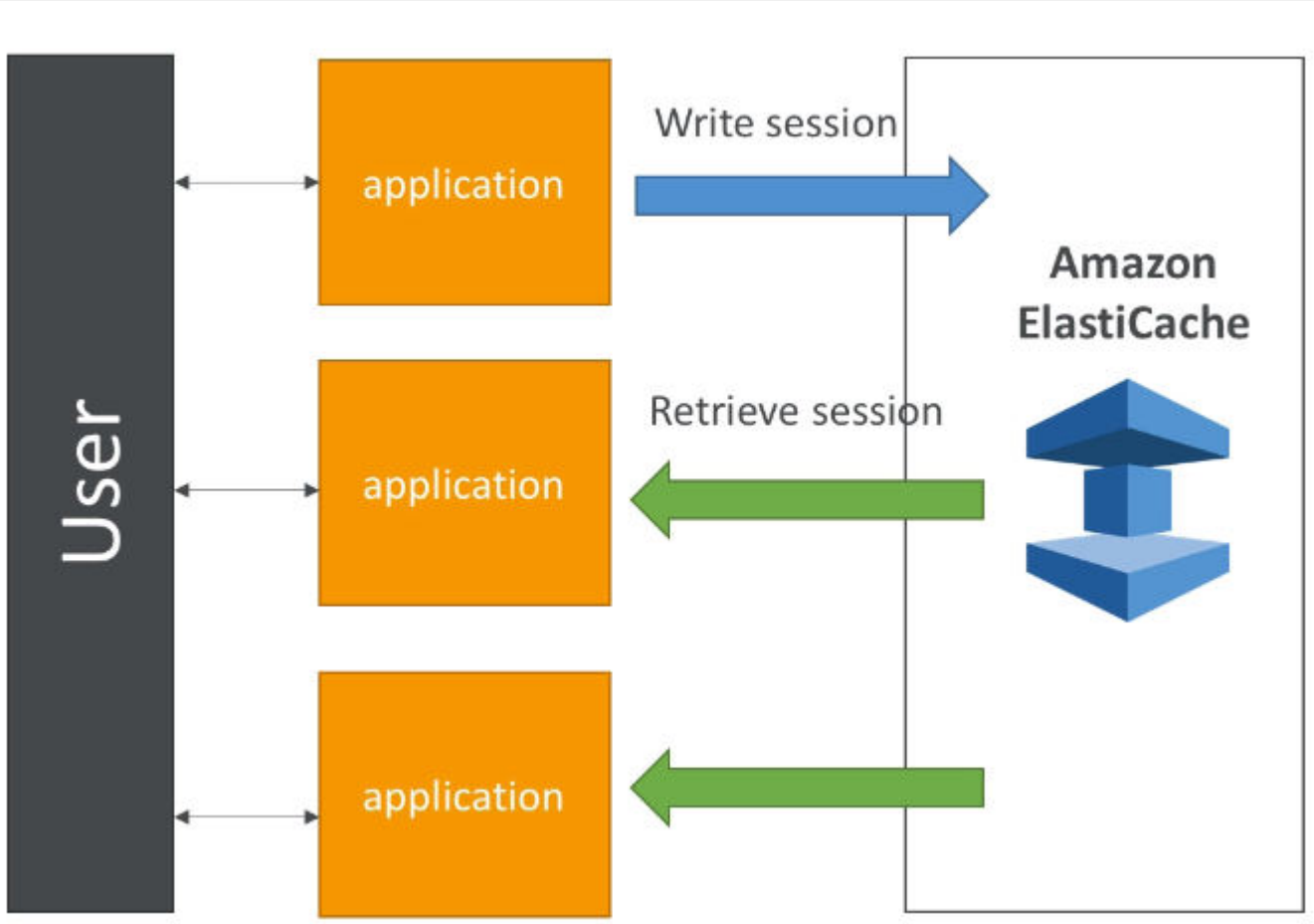
- Caching improves application performance by storing critical pieces of data in memory for low-latency access.
- Cached information may include the results of I/O-intensive database queries or the results of computationally intensive calculations.
- ElastiCache is a good choice if your database is particularly read heavy and not prone to frequent changing.

ElastiCache - DBCache



ElastiCache – User Session Store

- User logs into any of the application.
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



AWS has MemCached and Redis

MemCached & Redis

- **Memcached** is an open source, distributed, in-memory **key-value store-caching** system for small arbitrary data streams flowing from database calls, API calls, or page rendering. Memcached has long been the first choice of caching technology for users and developers around the world.

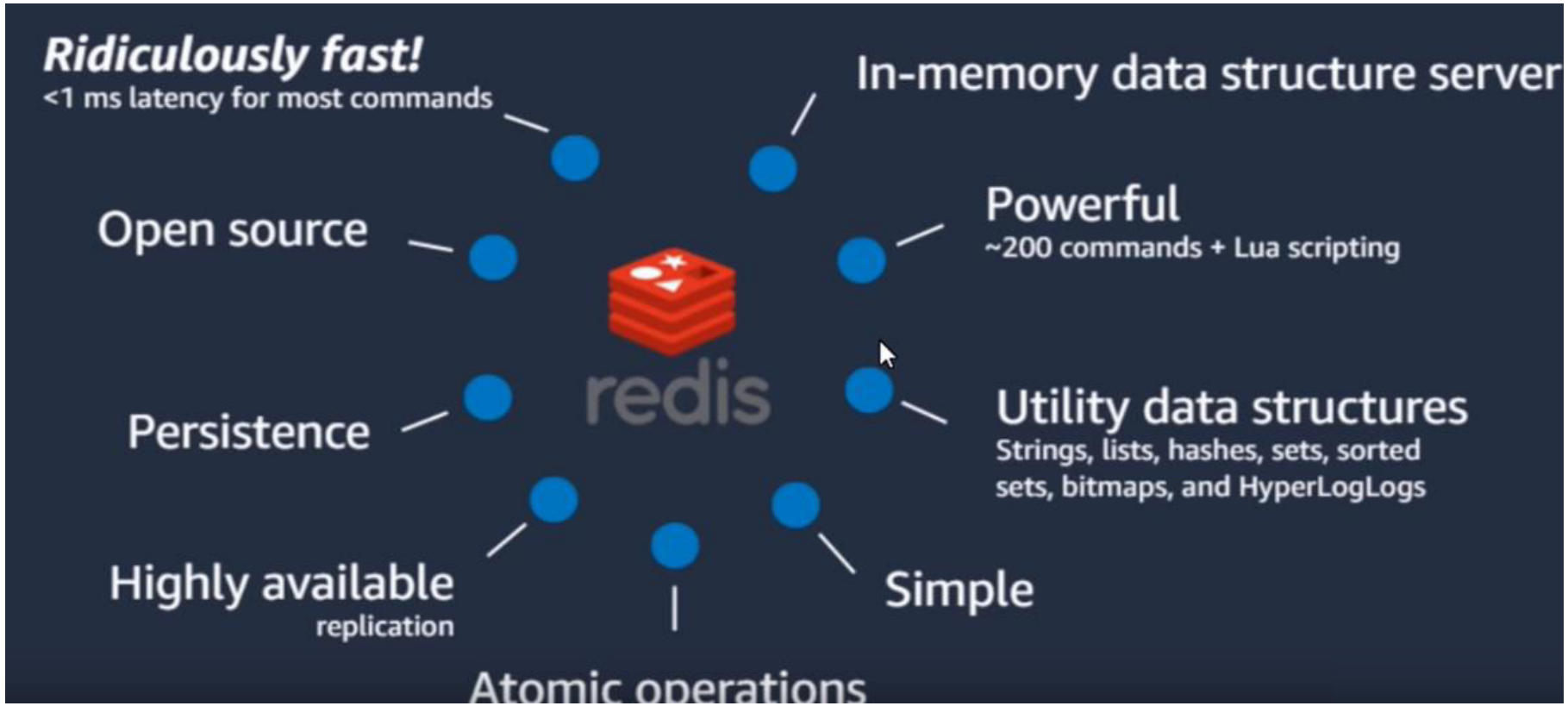
- **Redis** is a newer technology and often considered as a superset of Memcached. That means Redis offers more and performs better than Memcached. Redis scores over Memcached in few areas

MemCached & Redis

- **Memcached is a** High-performance, distributed memory object caching system, intended for use in speeding up dynamic web applications.

- **Redis** In-memory data structure store used as database, cache and message broker. ElastiCache for Redis **offers Multi-AZ with Auto-Failover** and enhanced robustness.

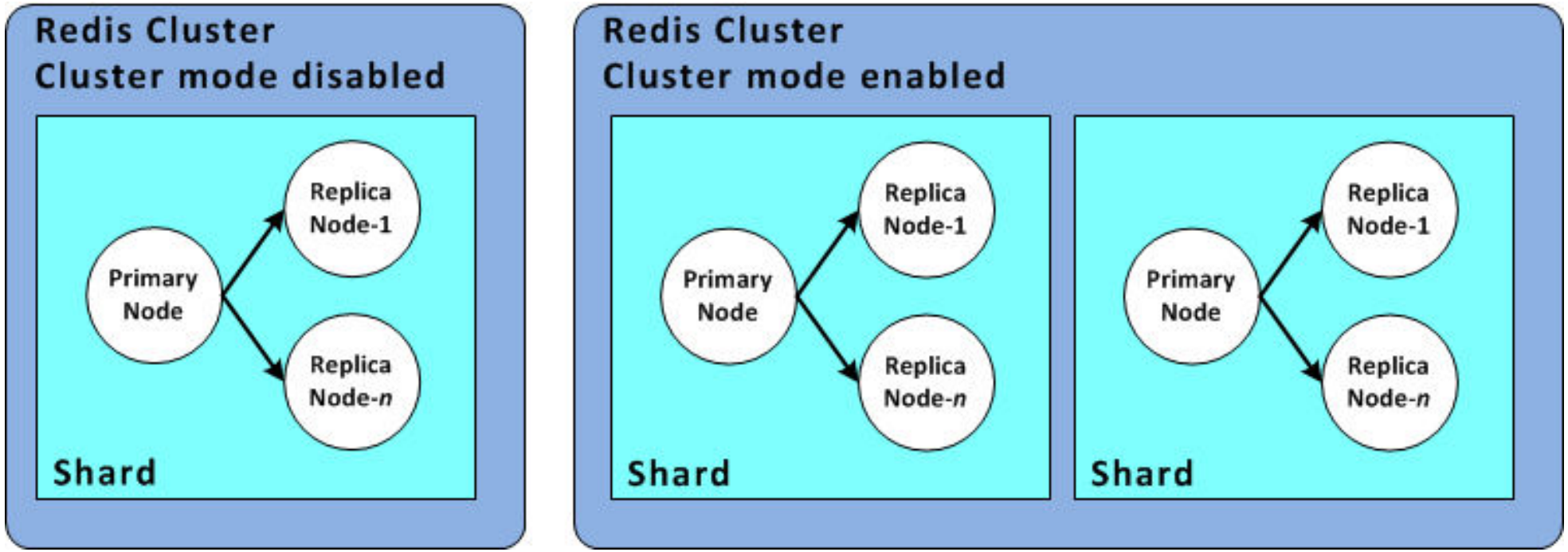
Redis


Ridiculously fast!
<1 ms latency for most commands
In-memory data structure server
Open source
Powerful
~200 commands + Lua scripting
redis
Persistence
Utility data structures
Strings, lists, hashes, sets, sorted sets, bitmaps, and HyperLogLogs
Highly available
replication
Simple
Atomic operations

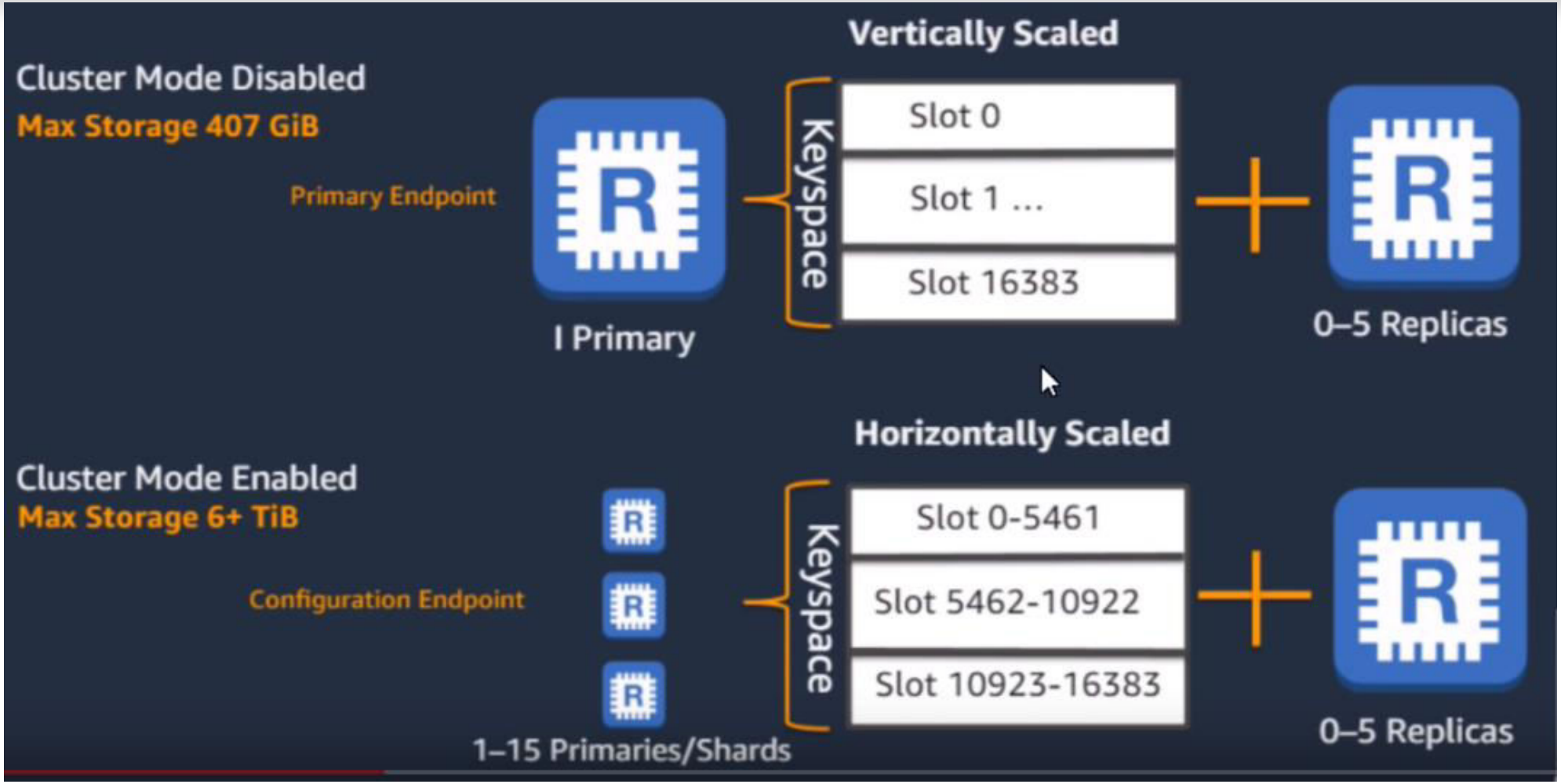
Redis: Shards

- A shard (API/CLI: node group) is a collection of one to six Redis nodes(1 primary + 5 replicas)
- A Redis (cluster mode disabled) cluster will never have more than one shard.
- Redis (cluster mode enabled) clusters can have from 1 to 90 shards
- You can create a cluster with higher number of shards and lower number of replicas totaling up to 90 nodes per cluster
- The node or shard limit can be increased to a maximum of 250 per cluster

Redis Topology



Redis Topology



Redis Cluster-mode enabled vs disabled

Feature	Enabled	Disabled
Failover	15–30 sec (Non-DNS)	~1.5 min (DNS-based)
Failover risk	• Writes affected—partial dataset (less risk with more partitions) • Reads available	• Writes affected on entire dataset • Reads available
Performance	Scales with cluster size (90 nodes—15 primaries + 0–5 replicas per shard)	6 nodes (1 primary + 0–5 replicas)
Max connections	• Primaries (65,000 x 15 = 975,000) • Replicas (65,000 x 75 = 4,875,000)	• Primary: 65,000 • Replicas: (65,000 x 5 = 325,000)
Storage	6+ TiB	407 GB
Cost	Smaller nodes but more $$	Larger nodes less $

Elasticache Encryption

Encryption

- In-Transit: encrypt all communications between clients and Redis server as well as between nodes
- At-Rest: encrypt backups on disk and in Amazon S3
- Fully managed: setup via API or console, automatic issuance and renewal

What is ElastiCache ?

- The same way RDS is to get managed Relation Databases..
- ElastiCache is to get managed Redis or Memcached.
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Write Scaling using Sharding
- Read Scaling using read Replicas
- Multi AZ with Failover Capability.
- AWS takes care of OS maintenance / Patching, optimizations, setup, configurations, monitoring, failure recovery and backups

Redis Overview

- Redis is an in-memory key-value store
- Super low latency (sub ms).
- Cache survive reboots by default (its called persistence)
- Great to host
  - User sessions
  - LeaderBoard (gaming)
  - Relieve pressure on DB (such as RDS)
  - Pub / Sub capability for messaging
- Multi AZ with Automatic Failover for DR if you don’t want to lose your cache data
- Support for Read Replicas

Redis

- Redis implements six fine-grained policies for purging old data, while Memcached uses the LRU (Least Recently Used) algorithm.
- Redis supports key names and values up to 512 MB, whereas Memcached supports only 1 MB.
- Redis uses a **hashmap** to store objects whereas Memcached uses **serialized strings**.
- Redis provides a persistence layer and supports complex types like hashes, lists (ordered collections, meant for queue), sets (unordered collections of non-repeating values), or sorted sets (ordered/ranked collections of non-repeating values).
- Redis is used for built-in pub/sub, transactions (with optimistic locking), and Lua scripting.
- Redis 3.0 supports clustering.

Elasticache Usage patterns

- Session Management
- Database Caching
- Streaming data analytics
- APIs
- IOT
- Social media
- Standalone databases
- Publisher / Subscriber

Elasticache Redis Usage patterns

#1 Key-Value Store*

Fast in-memory data store in the cloud. Use as a database, cache, message broker, queue

Highly Available & Reliable

Read replicas, multiple primaries, multi-AZ with automatic failover

Fully Managed & Hardened

Fast in-memory data store in the cloud. Use as a database, cache, message broker, queue

Easily Scalable

Cluster with up to 6.1 TiB of in-memory data

Read scaling with replicas

Write and memory scaling with sharding

Scale out or in

Secure & Compliant

VPC for cluster isolation, encryption at rest/transit, HIPAA compliance

Memcached Overview

- Memcached is an in-memory object store
- Cache doesn't survive reboots

Use cases

  - Quick retrieval of objects from memory
  - Overall, Redis has largely grown in popularity and has better feature sets than Memcached.

- I would personally only use Redis for caching needs.
- AWS exam wouldn’t ask if Redis or Memcached is better. Just ElastiCache in general.

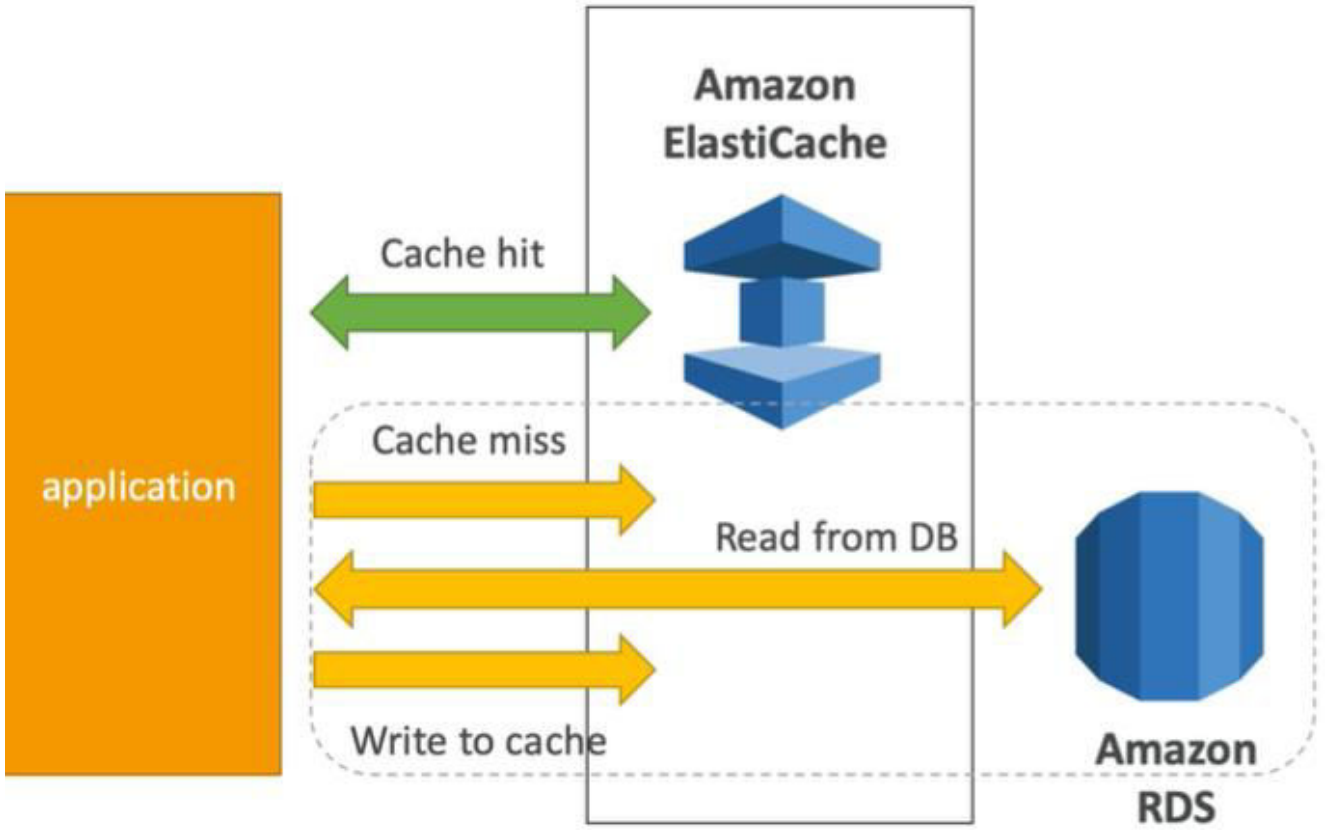
Memcached VS Redis

Requirement	Memcached	Redis
Simple Cache to offload DB	Yes	Yes
Ability to scale horizontally	Yes	No
Multi-threaded performance	Yes	No
Advanced data types	No	Yes
Ranking/Sorting data sets	No	Yes
Pub/Sub capabilities	No	Yes
Persistence	No	Yes
Multi-AZ	No	Yes
Backup & Restore Capabilities	No	Yes

Elasticache Patterns

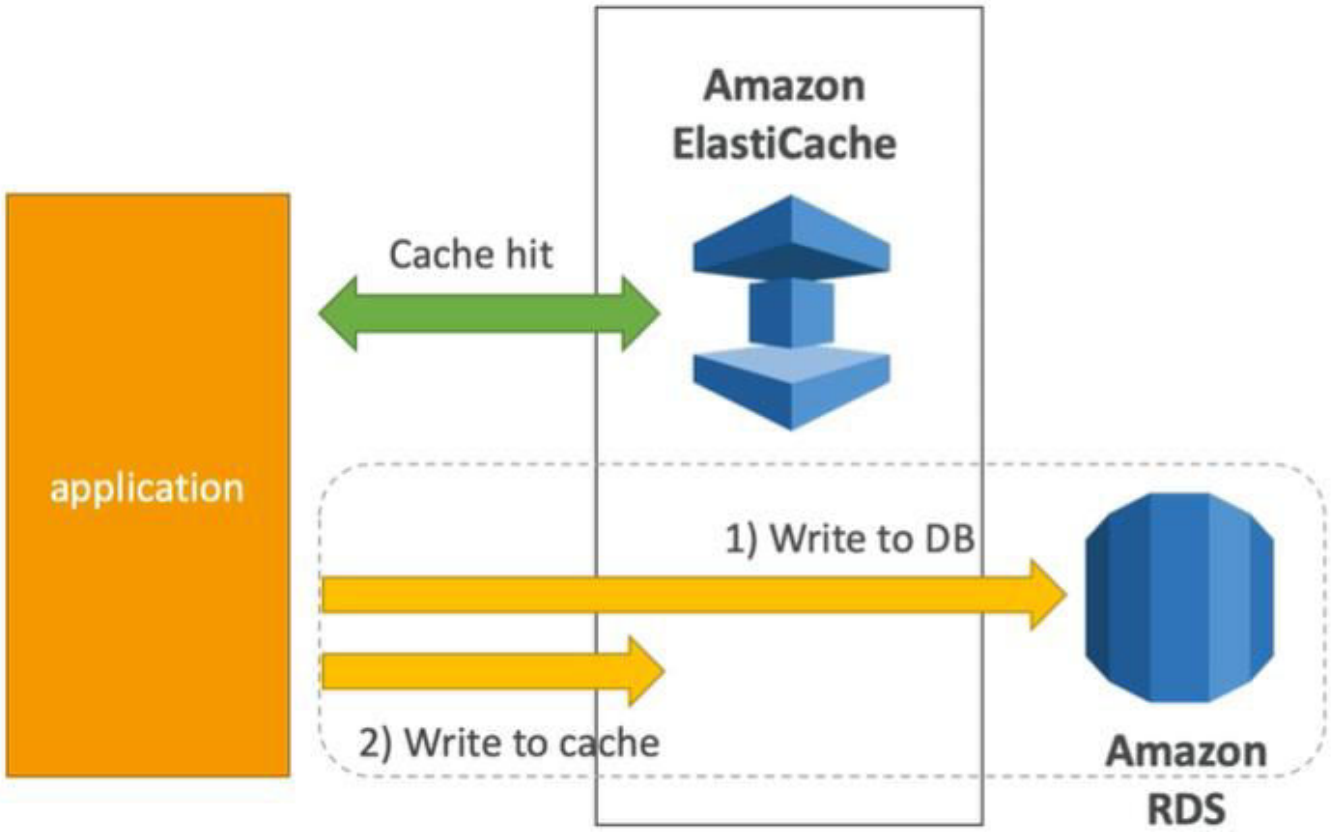
- ElastiCache is helpful for read-heavy application workloads
- There are two patterns / Cache strategies for ElastiCache
  - Lazy Loading
  - Write Through
- Strategies may be different based on the kind of application you have

Lazy Loading – Load only When necessary



- Pros
  - Only requested data is cached (the cache isn't filled up with unused data)
  - Node failures are not fatal (just increased latency to warm the cache)
- Cons
  - Cache miss penalty that results in 3 round trips, noticeable delay for that request
  - Stale data: data can be updated in the database and outdated in the cache

Write Through- Add or Update when database is update



- Pros:
  - Data in cache is never stale
  - Write penalty vs Read penalty (each write requires 2 calls)
- Cons:
  - Missing Data until it is added / updated in the DB. Mitigation is to implement Lazy Loading strategy as well
  - Cache churn – a lot of the data will never be read

Exam Tips

- Use ElastiCache to increase database and web application performance.
- Redis is Multi-AZ
- You can do backups and restored of Redis
- If you need to scale horizontally use Memcached.
- ElastiCache manage applications session state.
- Typically you will be given a scenario where a particular database is under lot of stress/load . You may be asked which service you should use

ElastiCache is a good answer if your database is particularly read heavy and not prone to frequent changes