Elasticache

A CLOUD GURU

- Elasticache - in memory cache in the cloud
- Improves performance of web applications, allowing you to retrieve information from fast in-memory caches rather than slower disk based databases
- Sits between your application and the database:
  - e.g. an application frequently requesting specific product information for your best selling products
- Takes the load off your databases
- Good if your database is particularly **read-heavy** and the data is not changing frequently

Elasticache Benefits and Use Cases

A CLOUD GURU

- Improves performance for read-heavy workloads
  - e.g. Social Networking, gaming media sharing, Q&A portals
- Frequently-accessed data is stored in memory for low-latency access, improving the overall performance of your application.
- Also good for compute heavy workloads
  - e.g. recommendation engines
- Can be used to store results of I/O intensive database queries or output of compute-intensive calculations.

2 Types of Elasticache

A CLOUD GURU

- Memcached
  - Widely adopted memory object caching system
  - Muti-threaded
  - No Multi-AZ capability

- Redis
  - Open-source in-memory key-value store
  - Supports more complex data structures: sorted sets and lists
  - Supports Master / Slave replication and Multi-AZ for cross AZ redundancy

Caching Strategies

A CLOUD GURU

- 2 strategies available: Lazy Loading and Write-Through:

  - Lazy Loading - Loads the data into the cache only when necessary
  - If requested data is in the cache, Elasticache returns the data to the application.
  - If the data is not in the cache or has expired, Elasticache returns a `null`.
  - Your application then fetches the data from the database and writes the data received into the cache so that it is available next time.

Lazy Loading Advantages and Disadvantages

A CLOUD GURU

Advantages	Disadvantages
Only requested data cached: Avoids filling up cache with useless data	Cache miss penalty : Initial request Query to database Writing of data to the cache
Node failures are not fatal a new empty node will just have a lot of cache misses initially	Stale data - if data is only updated when there is a cache miss, it can become stale. Doesn't automatically update if the data in the database changes

Lazy Loading and TTL

- TTL (Time To Live)
  - Specifies the number of seconds until the key (data) expires to avoid keeping stale data in the cache
  - Lazy Loading treats an expired key as a cache miss and causes the application to retrieve the data from the database and subsequently write the data into the cache with a new TTL
  - Does not eliminate stale data — but helps to avoid it

Write-Through Advantages and Disadvantages

A CLOUD GURU

- Write Through - adds or updates data to the cache whenever data is written to the database

Disdvantages	Advantages
Write penalty : Every write involves a write to the cache as well as a write to the database	Data in the cache never stale
If a node fails and a new one is spun up, data is missing until added or updated in the database (mitigate by implementing Lazy Loading in conjunction with write-through)	Users are generally more tolerant of additional latency when updating data than when retrieving it
Wasted resources if most of the data is never read	

Elasticache Exam Tips

A CLOUD GURU

- In-memory cache sits between your application and database
- 2 different caching strategies: Lazy loading and Write Through
- Lazy Loading only caches the data when it is requested
- Elasticache Node failures not fatal, just lots of cache misses
- Cache miss penalty: Initial request, query database, writing to cache
- Avoid stale data by implementing a TTL

Elasticache Exam Tips

A CLOUD GURU

- Write Through strategy writes data into the cache whenever there is a change to the database
- Data is never stale
- Write penalty: Each write involves a write to the cache
- Elasticache node failure means that data is missing until added or updated in the database
- Wasted resources if most of the data is never used