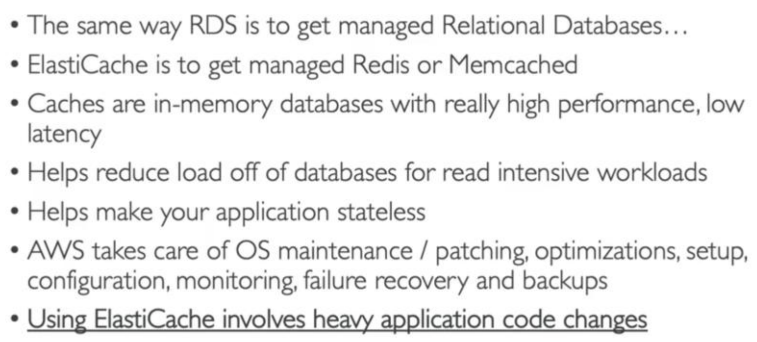
Amazon ElastiCache Overview

AWS ElastiCache is a fully managed, in-memory caching service provided by Amazon Web Services (AWS). It is designed to improve the performance and scalability of applications by providing a fast, scalable, and secure in-memory data store.

Key features and aspects of AWS ElastiCache include:

1. **Caching Solution:** ElastiCache is primarily used as a caching layer to reduce the load on backend databases by storing frequently accessed data in-memory. It helps improve application performance by reducing database load times and improving response times.
2. **In-Memory Data Stores:** ElastiCache supports two popular in-memory data stores:
    - **Redis:** A versatile, open-source, and high-performance in-memory data store that supports advanced data structures like strings, lists, sets, sorted sets, hashes, etc. It is commonly used for caching, real-time analytics, and session management.
    - **Memcached:** A simple, fast, and distributed in-memory caching system that stores data in key-value pairs. It is suitable for caching frequently accessed data and improving application performance.
3. **Managed Service:** AWS ElastiCache is a fully managed service, which means AWS handles the infrastructure provisioning, patching, maintenance, and backups. Users can focus on using the caching service without managing the underlying infrastructure.
4. **Scalability:** ElastiCache allows users to easily scale their cache clusters vertically by modifying node types or horizontally by adding or removing nodes to accommodate varying workload demands.
5. **High Availability:** It supports features like Multi-AZ with Redis, which provides high availability and failover protection across different Availability Zones, ensuring data durability and minimal downtime in case of node failures.
6. **Security:** ElastiCache integrates with AWS Identity and Access Management (IAM) for fine-grained access control. It also supports encryption at rest and in transit to enhance data security.
7. **Compatibility:** ElastiCache seamlessly integrates with other AWS services like Amazon RDS, Amazon EC2, AWS Lambda, and more, allowing applications to leverage caching capabilities easily.

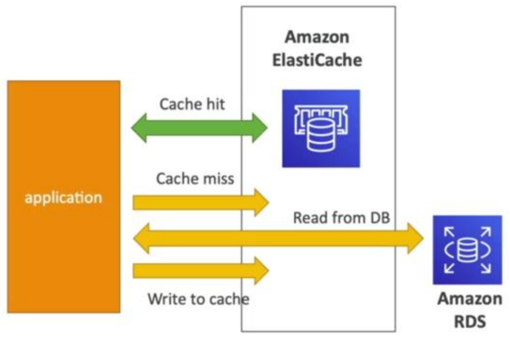
Overall, AWS ElastiCache is a valuable tool for optimizing application performance by providing fast and scalable in-memory caching solutions, reducing latency and improving the overall user experience for applications that rely on frequently accessed data.



- The same way RDS is to get managed Relational 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
- Helps make your application stateless
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups
- Using ElastiCache involves heavy application code changes

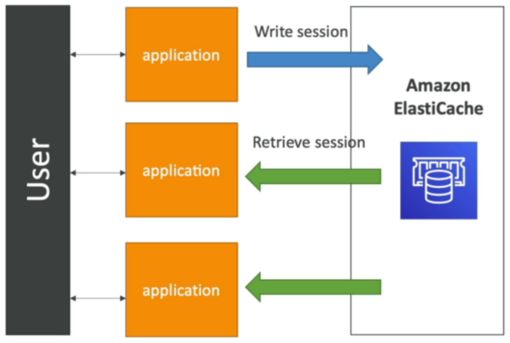
ElastiCache Solution Architecture – DB Cache

- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.



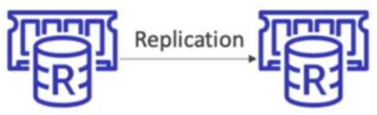
ElastiCache Solution Architecture – User Session Store

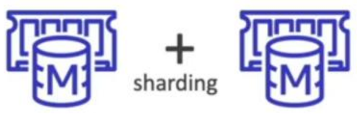
- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



ElastiCache – Redis vs Memcached

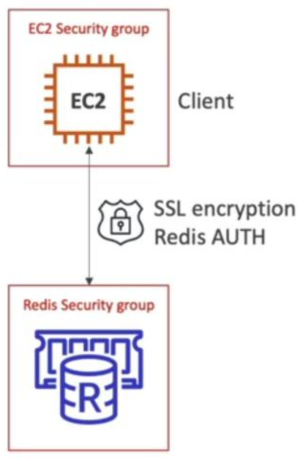
REDIS	MEMCACHED
• Multi AZ with Auto-Failover	• Multi-node for partitioning of data (sharding)
• Read Replicas to scale reads and have high availability	• No high availability (replication)
• Data Durability using AOF persistence	• Non persistent
• Backup and restore features	• No backup and restore
• Supports Sets and Sorted Sets	• Multi-threaded architecture





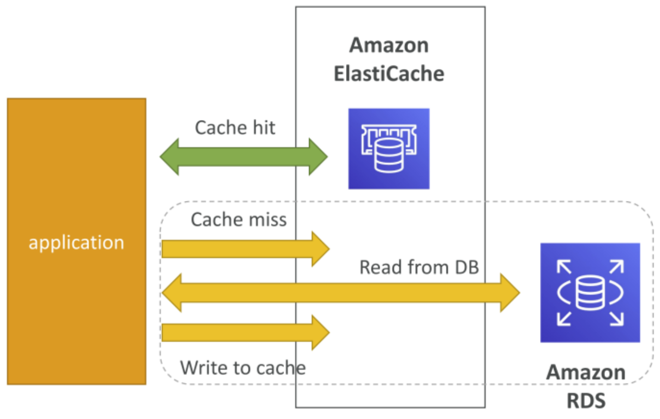
ElastiCache Cache Security

- ElastiCache supports **IAM Authentication for Redis**
- IAM policies on ElastiCache are only used for AWS API-level security
- Redis AUTH
  - You can set a "password/token" when you create a Redis cluster
  - This is an extra level of security for your cache (on top of security groups)
  - Support SSL in flight encryption
- Memcached
  - Supports SASL-based authentication (advanced)



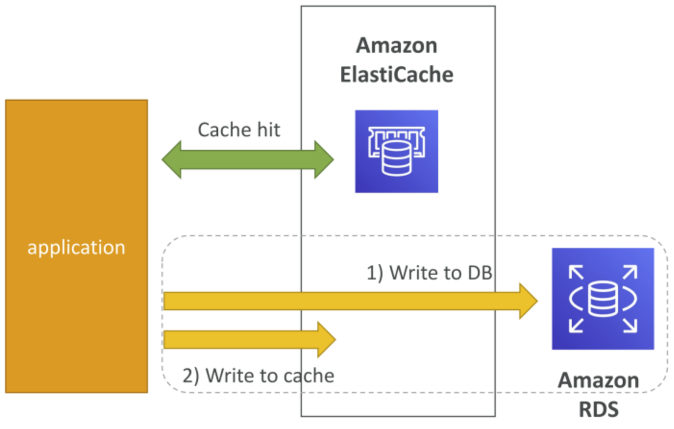
Patterns for ElastiCache

Lazy Loading / Cache-Aside / Lazy Population



- 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 cache when database is updated:



- Pros:
  - Data in cache is never stale, **reads are quick**
  - **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

Cache Evictions and Time-to-live (TTL)

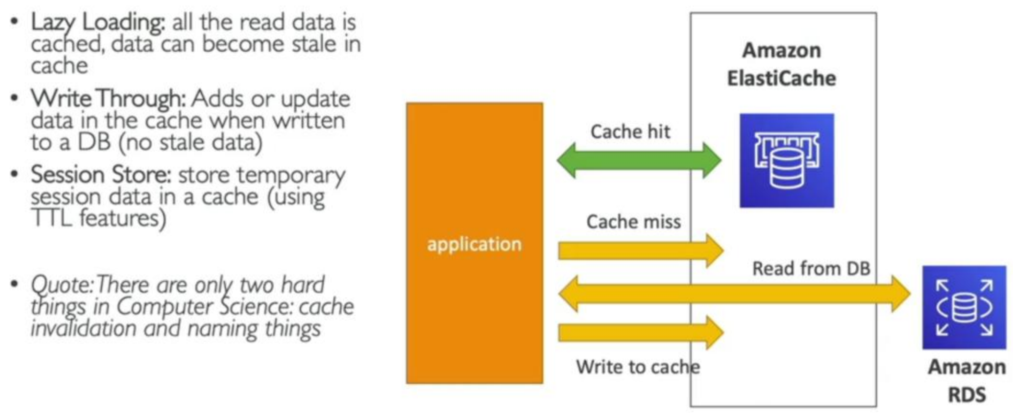
- Cache eviction can occur in three ways:
  - You delete the item explicitly in the cache
  - Item is evicted because the memory is full and it's not recently used (LRU)
  - You set an item **time-to-live (or TTL)**
- TTL are helpful for any kind of data:
  - Leaderboards
  - Comments
  - Activity streams
- TTL can range from few seconds to hours or days

- If too many evictions happen due to memory, you should scale up or out

When we need to use which Patterns of ElastiCache

- Lazy Loading / Cache aside is easy to implement and works for many situations as a foundation, especially on the read side
- Write-through is usually combined with Lazy Loading as targeted for the queries or workloads that benefit from this optimization
- Setting a TTL is usually not a bad idea, except when you're using Write-through. Set it to a sensible value for your application
- Only cache the data that makes sense (user profiles, blogs, etc…)

Patterns for ElastiCache

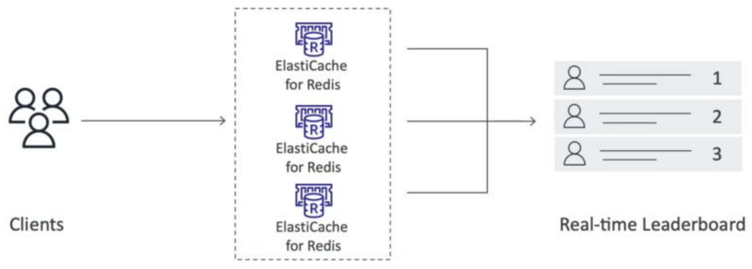


- Lazy Loading: all the read data is cached, data can become stale in cache
- Write Through: Adds or update data in the cache when written to a DB (no stale data)
- Session Store: store temporary session data in a cache (using TTL features)
- *Quote: There are only two hard things in Computer Science: cache invalidation and naming things*

Lazy Loading illustrated

ElastiCache – Redis Use case:

- Gaming Leaderboards are computationally complex
- Redis Sorted sets guarantee both uniqueness and element ordering
- Each time a new element added, it's ranked in real time, then added in correct order



Important ports:

- FTP: 21
- SSH: 22
- SFTP: 22 (same as SSH)
- HTTP: 80
- HTTPS: 443

RDS Databases ports:

- PostgreSQL: 5432
- MySQL: 3306
- Oracle RDS: 1521
- MSSQL Server: 1433

- MariaDB: 3306 (same as MySQL)
- Aurora: 5432 (if PostgreSQL compatible) or 3306 (if MySQL compatible)