Elastic Cache
Amazon ElastiCache

ElastiCache is an in-memory hosted caching solution provided by AWS. ElastiCache supports two types of caching engines at this time.

Memcached – Common caching appliance which uses a DB source such as mariadb/MySQL as the persistent storage and fills frequently accessed objects inside of in-memory memcache.

Redis – Redis acts more like a replacement for the DB server and instead maintains its own persistence and is used for certain types of application functions.

*Note: Each caching engine provides different methods for high availability, backup, usage, and migration.*
Amazon ElastiCache: When to cache?

Cache data that is “static” and is also frequently accessed
  • Profile data

Storing infrequently accessed data doesn’t equate to cost savings or much performance savings but will fill up your available cache memory.

Cache expensive queries or slow queries with joins that run across multiple tables, these are considered hardware intensive and expensive.

Cache data is “stale” it doesn’t change frequently and would require flushing for new data to appear.
  • Redis caching engine is a little different as it uses the in-memory storage for actual data storage and only writes persistence to snapshots or data files frequently.
Amazon ElastiCache: When to cache?

Is the query being made against the database slow or expensive?
• Large join showing the results of comments on a wordpress thread

Is the resulting data frequently accessed?
• Social media profile or even a course listing
Amazon ElastiCache: Caching Strategies

Lazy Loading
- Application attempts to receive data from the cache nodes
- If no data is available then the cache nodes return null
- Application receives the data from the database (disk based db)
- Application then updates the cache
- Only requested data is cache, so the cache is not filling up the memory with non-requested data and taking resources
- Node failures aren’t a huge issue because if a node fails the request just goes to the DB

Lazy loading can be expensive if there is a cache miss. This is important in determining if an item is infrequently accessed and should be cache or not. If it is infrequently accessed it will be less expensive to just read from the DB and bypass cache.
Amazon ElastiCache: Caching Strategies

- Ensures data is never stale and is always up to date (does not require expiration)
- Each DB write involves two steps, write to db and write to cache can become expensive by increasing latency
  - Good strategy for applications that do not have a lot of writes

- Downsides:
  - Lots of data is stored in memory that may not be frequently accessed
  - If a node is spinning up it could miss writing and cause missing data
Amazon ElastiCache: Caching Strategies

Adding TTL: Essentially, cache expires after the TTL (Time To Live) which can be applied to both lazy loading and write through to manage cache resources.

- Number of seconds until a key expires (caching is a key:value store)

Note: Anytime you access data from in-memory storage, it is ephemeral but is MUCH faster than reading from a disk. Remember, the type of data you cache depends on the caching engine you are using, the use case, and what it takes to load the data into cache.
Amazon ElastiCache: Memcached

Memcached is a more traditional caching mechanism which is placed in front of a DB source.

- Does not manage its own persistence
- Can be run in a cluster of nodes
- Does not have backup abilities
- Scales by adding more nodes to the cluster

Populate cache:
- Write through
- Lazy loading
Amazon ElastiCache: Memcache Lazy loading example
Amazon ElastiCache: Memcache
Amazon ElastiCache: Memcached

- If you need to scale the nodes in a cluster up or down to a different instance type, you must create a new cluster with the new node instance type
- Purchase reserved nodes to reduce costs -> not good for spot
- Can scale by adding on-demand nodes for times of increase in demand
- Every node in the cluster is the same instance type
- Memcached supports auto discovery, client programs automatically identify all nodes in a cache cluster
- Improve fault tolerance by locating nodes in multiple availability zones
Amazon ElastiCache: Memcached

- Memcached is a region only service there is no method for “migrating” ElastiCache clusters to another region other than firing up a new cluster and letting it populate in another region.

- In a multi-region design, have an ElastiCache cluster in each region populating data from the local/regional DB server.

- Memcached is a great solution for storing “session” state in applications this will make web servers stateless which allows for easily scaling.
Amazon ElastiCache: Memcached Backups

Memcached uses a database as its persistent storage in the event of a node failure cache misses will make requests to the backend DB to populate the cache engine.

Note: This can cause an increase load on your SQL server to mitigate this load use more nodes in a cluster so a loss of a node does not equate to a substantial increase in database load on your backend database store.

When “events” occur to clusters notifications can be configured to be sent to SNS topics for automation and notification
Amazon ElastiCache: Redis

Redis caching engine is substantially different than memcached. Redis provides persistent storage options instead of using a DB such as MySQL or MariaDB.

Redis uses:

- Small enough data sets that can be stored in-memory
- Need a persistent key store or caching engine that provides persistence
- Automatic failover to a backup node in case of node failure
- Backup and restore capabilities
- Leaderboards
- Data with intense calculations and frequent changing data
Amazon ElastiCache: Redis persistence

Redis is often used as a replacement of some DB servers which in a memcached cluster are what allows for persistence. To apply persistence to a cluster in the event of a reboot, enable Redis Append Only Files (AOF)

- Disabled by default
- Will write all commands that change cache to an “append-only” file
- If a node is rebooted and the memory is cleared then when Redis caching engine starts the AOF is loaded through the commands in the AOF file and the cache is available again.
Amazon ElastiCache: Redis

Scaling Redis

- Scales similar to RDS scaling to increase capacity for writes you need to increase instance size
- Redis also supports clusters of read replica groups
- To increase the size of a Redis node
  - Take a snapshot of the node
  - Launch a new instance with instance type based off of the snapshot
  - Can also launch a new cluster and “seed” it from a snapshot
Amazon ElastiCache: Redis backups

Redis is the only caching engine “currently” that supports backups on ElastiCache.

Automatic Snapshots — Backups are taken on a daily basis, select a snapshot window and time limit, if failure occurs on a cluster then the cluster can be restored from the most recent snapshot.

Manual Snapshots — Can be taken at anytime and are not subject to the “retention limit” of automatic backups.

Snapshots can be exported into an EC2 managed environment.

Redis snapshots can be copied but cannot be copied to another region they can only be “copied”.

QUALITY THOUGHT
INFO SYSTEMS (INDIA) PVT. LTD
Amazon ElastiCache: Use cases

- Leader boards
- Session state data
- Recommendation data
- Hootsuite session state example and why it’s good for failover