The Redis Architecture: Memory-first persistence models, supported data structures, and distributed caching roles.

---

1) What Redis Is For

- High-speed layer for APIs, rate limits, sessions, queues/streams, leaderboards.
- Speed comes from RAM-first design and purpose-built data structures.

2) Memory-First Architecture

- Everything lives in RAM; disk is only for recovery.
- Effects: sub-ms ops, no disk waits, predictable latency, tunable durability.

3) Persistence Options

- RDB snapshots: periodic full dumps; tiny files, quick restart; may lose seconds/minutes.
- AOF: append every write; minimal loss; larger files, rewrite cost.
- Hybrid (RDB + AOF tail): fast boot with near-latest data.

4) Typical Roles

- Pure cache: no persistence; rebuild after restart.
- Warm cache: light RDB or AOF-every-second to dodge cold start.
- Semi-durable store: AOF + rewrite for queues, counters, important state.

5) Execution Model

- Single-core command path; network/persistence on helper threads.
- Scale by more instances, not more threads on one node.

6) Scaling

- Vertical: add CPU/RAM; simplest; works while dataset fits one box.
- Horizontal:
    - Replication: primary + replicas for reads and HA.
    - Cluster (sharding): split keys across nodes when memory/traffic exceed one machine.

7) Caching Patterns

- Cache-aside (common): app checks cache, on miss fetches DB/API, then fills cache.
- Read-through: cache fetches on miss automatically.
- Write-through: write cache, cache writes DB; consistent but slower writes.
- Write-back: write cache, flush later; fastest writes, risk if node dies before flush.

8) Core Data Structures

- Strings (counters/tokens), Hashes (objects), Lists (queues), Sets (unique tags), Sorted Sets (ranks/windows), Streams (event logs with consumer groups), Bitmaps/HyperLogLog (large-scale analytics).