



Mayank Ahuja • Following

Follow for Your Daily Dose of AI, Software Development & System Design Tips | Exploring AI SaaS - Tink...

[View my blog](#)

19h • 🌐

Let's understand - Redis

Do you know?

➡➡ Redis wasn't born in a lab, it was born from frustration. 😊

How?

➡➡ Back in 2009, Salvatore Sanfilippo needed a faster way to analyze website traffic for his startup.

Existing databases just couldn't keep up.

➡➡ So, he built his own solution => Redis (a super-fast data store that lives in your computer's memory.)

It was a game-changer, and it's still transforming how we handle data today.

📌 **Let's understand in detail.**

Redis => REmote DIctionary Server

- initially designed to be a remote data structure server accessible over the network.
- open-source, in-memory data store.
- functions mainly as a key-value database
- but goes beyond by offering a variety of data structures like strings, lists, sets, sorted sets, hashes etc.
- primarily resides in RAM, providing incredibly fast read and write operations.
- options for persistence to disk, ensuring data durability.

Redis is fast. Why?

As we discussed,

Unlike traditional databases that primarily store data on disk, Redis keeps its entire dataset in memory (RAM).

This eliminates the latency associated with disk seeks and reads, allowing for very fast data access.

📌 **RESP Protocol** => REdis Serialization Protocol

- Redis uses its own binary-safe protocol called RESP for communication.
- designed for simplicity and efficiency
- less overhead as compared to text-based protocols like HTTP

📌 **Event Loop and I/O Multiplexing**

- The heart of Redis => event loop. (single-threaded)
- continuously monitors file descriptors (sockets) for client connections and incoming commands.
- event-driven architecture, combined with I/O multiplexing (epoll, kqueue, or select) => allows Redis to handle thousands of concurrent clients efficiently without the overhead of multiple threads.
- non-blocking I/O operations
- not waiting for I/O operations to complete
- main thread remains responsive and can quickly process other commands
- if main thread encounters a time-consuming operation (like accessing the disk or network), it doesn't halt and wait => delegates the task to the operating system and registers a callback function to be executed once the operation completes.

📌 **Memory Fragmentation and jemalloc**

- Redis uses a memory allocator called jemalloc to manage memory efficiently.
- Jemalloc helps reduce memory fragmentation, which can occur when objects are allocated and freed repeatedly.

📌 **Redis offers various deployment modes.**


- Standalone - A single instance of Redis.

- Cluster - A distributed implementation for scalability and high availability.
- Sentinel - High availability for standalone or replicated Redis instances.
- Replication - Master-replica setup for data redundancy and read scalability.

👍 Follow - [Mayank Ahuja](#)

📧 Newsletter - <https://lnkd.in/dJByxEYY>

#softwaredevelopment #redis #databases



UNDERSTANDING REDIS

IN-MEMORY DATA STORE

REDIS WASN'T BORN IN A LAB - IT WAS BORN FROM FRUSTRATION. 😞

Back in 2009, Salvatore Sanfilippo needed a faster way to analyze website traffic for his startup. Existing databases just couldn't keep up.

So, he built his own solution => Redis (a super-fast data store that lives in your computer's memory.)

It was a game-changer, and it's still transforming how we handle data today.



REDIS => REMOTE DICTIONARY SERVER

- initially designed to be a remote data structure server accessible over the network.
- open-source, in-memory data store.
- functions mainly as a key-value database
- but goes beyond by offering a variety of data structures like strings, lists, sets, sorted sets, hashes etc.
- primarily resides in RAM, providing incredibly fast read and write operations.
- options for persistence to disk, ensuring data durability.




REDIS IS FAST

Unlike traditional databases that primarily store data on disk, Redis keeps its entire dataset in memory (RAM). This eliminates the latency associated with disk seeks and reads, allowing for very fast data access.

RESP Protocol => Redis Serialization Protocol

- Redis uses its own binary-safe protocol called RESP for communication.
- designed for simplicity and efficiency
- less overhead as compared to text-based protocols like HTTP



EVENT LOOP & I/O MULTIPLEXING

- The heart of Redis => event loop. (single-threaded)
- continuously monitors file descriptors (sockets) for client connections and incoming commands.
- event-driven architecture, combined with I/O multiplexing (epoll, kqueue, or select) => allows Redis to handle thousands of concurrent clients efficiently without the overhead of multiple threads.
- non-blocking I/O operations
- not waiting for I/O operations to complete
- main thread remains responsive and can quickly process other commands
- if main thread encounters a time-consuming operation (like accessing the disk or network), it doesn't halt and wait => delegates the task to the operating system and registers a callback function to be executed once the operation completes.



Memory Fragmentation and jemalloc

Redis uses a memory allocator called jemalloc to manage memory efficiently.

- Jemalloc helps reduce memory fragmentation, which can occur when objects are allocated and freed repeatedly.

