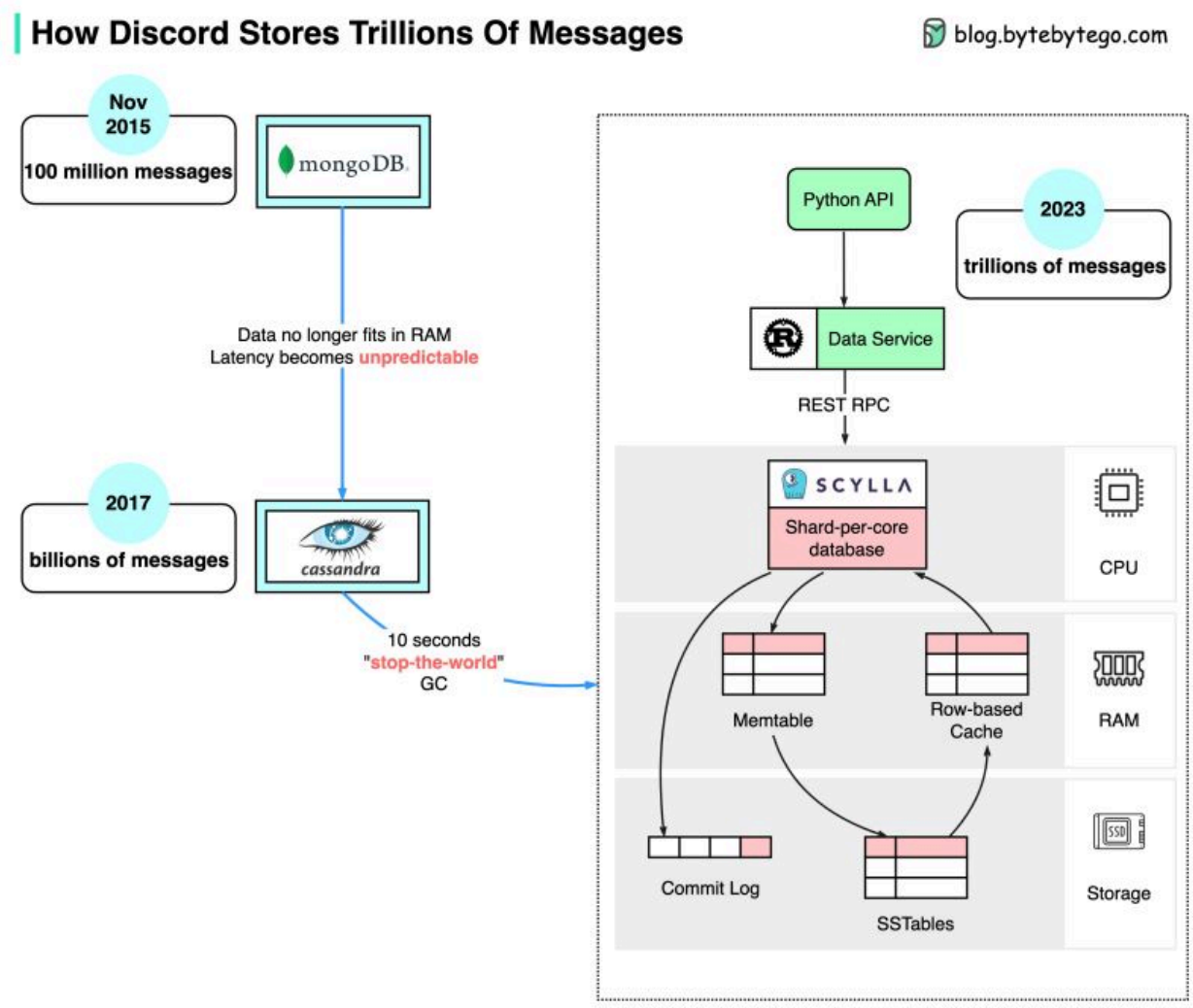
How Discord Stores Trillions Of Messages

The diagram below shows the evolution of message storage at Discord:

MongoDB ➡️ Cassandra ➡️ ScyllaDB



In 2015, the first version of Discord was built on top of a single MongoDB replica. Around Nov 2015, MongoDB stored 100 million messages and the RAM couldn't hold the data and index any longer. The latency became unpredictable. Message storage needs to be moved to another database. Cassandra was chosen.

In 2017, Discord had 12 Cassandra nodes and stored billions of messages.

At the beginning of 2022, it had 177 nodes with trillions of messages. At this point, latency was unpredictable, and maintenance operations became too expensive to run.

There are several reasons for the issue:

- Cassandra uses the LSM tree for the internal data structure. The reads are more expensive than the writes. There can be many concurrent reads on a server with hundreds of users, resulting in hotspots.
- Maintaining clusters, such as compacting SSTables, impacts performance.
- Garbage collection pauses would cause significant latency spikes

ScyllaDB is a Cassandra compatible database written in C++. Discord redesigned its architecture to have a monolithic API, a data service written in Rust, and ScyllaDB-based storage.

The p99 read latency in ScyllaDB is 15ms compared to 40-125ms in Cassandra. The p99 write latency is 5ms compared to 5-70ms in Cassandra.

Over to you: What kind of NoSQL database have you used? How do you like it?

References:

- Shards per core architecture
- How discord stores trillions of messages