NoSQL

Benmounah Zakaria
Constantine 2 University
SDIA M1
2023-2024

PART 01

HISTORY ERA BEFORE NOSQL

Pre-2002 Era

- Most organizations had their own relational databases.
- They would buy expensive servers (e.g., $40,000 computers) to host and manage their databases. This was referred to as vertical scaling.
- Some vendors claimed to be "cloud" by merely collecting individual databases and hosting them on platforms like Amazon. This wasn't true cloud computing but more of a collection of individual databases.

Web Search and Crawling

Late 1990s - Early 2000s

- Google set out to index the entire internet.
- Traditional databases couldn't handle such a vast amount of data.
- Google's solution was to crawl the web, make a full copy, and then index that copy.
- This endeavor required new data management and storage techniques, as the scale was unlike anything seen before in the database world.

Gmail Introduction
April 1, 2004

- When Google introduced Gmail, it wasn't just a new email service; it was a challenge to the established norms of data management.
- Traditional relational databases were designed for smaller, isolated workloads.
- In contrast, Gmail had to serve millions of users globally, providing each with a significant amount of storage (1 GB at launch, which was unprecedented at the time).
- Gmail's architecture needed to be highly distributed and scalable. Each user's data had to be isolated, yet the system had to be responsive and reliable.

Data Silos and Eventual Consistency

2000s onwards

- Google's applications, including Gmail, were designed to operate with **data silos**.
- Each user's interactions were restricted to their own "silo" or segment of data.
- For instance, if a user deleted an email, it affected only their silo and not others.
- This allowed for "eventual consistency," where data didn't need to be immediately consistent across all servers but would eventually reach a consistent state.
- This approach was different from traditional databases, which prioritized immediate consistency (ACID properties).

Dynamic Data Migration
2000s onwards

- In the early days of Gmail, Google had the capability to migrate a user's data closer to their geographical location.
- If a user traveled from the US to Europe, their data might shift to European servers, making access faster.
- This dynamic migration was innovative and showcased Google's commitment to user experience.
- Google's approach to handling large-scale, user-centric applications required rethinking traditional database architectures.

Revelations from Google

- In conferences like Google I/O 2008, Google began to share insights into how they built their systems. These revelations were revolutionary for many in the tech industry.
- The speaker was particularly impressed by Google's gather/scatter technique, where Google used numerous cheaper computers to achieve faster performance than one could get from a single expensive machine.
- By 2010, Google started showcasing more of their techniques, emphasizing efficiency and sustainability. They recognized the importance of sharing best practices for the greater good.

Observe where the story first started...

- https://www.youtube.com/watch?v=Md7K90FfJhg
- https://www.youtube.com/watch?v=Ho1GEyftpmQ
- https://www.youtube.com/watch?app=desktop&v=6x0cAzQ7PVs

Marissa Mayer Video

- Google search's perceived intelligence operates within a timeframe that's indistinguishable for users,
- whether it takes a tenth of a second or a thousandth of a second.
- The scatter-gather approach is efficient even if it takes a quarter of a second, as the resources used during that time are minuscule.
- This approach ensures consistency in search speed no matter the search volume.

Container Tour Video

- Google's data centers are built for efficiency, not aesthetics.
- There's no unnecessary packaging or wasted energy.
- The computers are designed to be disposable and modular, emphasizing replication to ensure data security.
- Each user's Gmail data, for example, is replicated multiple times across various servers and locations.

Matt Cutts Video

- Google's approach to computing **page rank** is elegantly simple when understood.
- It is a distributed algorithm, designed to converge and adjust as new pages emerge.
- Google's approach is unique, opting for distributed small databases rather than one large centralized one.

Shift to Amazon & AWS

- Amazon's evolution from a book company to a tech giant with AWS changed the tech landscape.
- Instead of investing in expensive hardware that becomes obsolete, the idea of renting space and computational power on cheap, commodity hardware became appealing.
- This led to the rise of "carpet clusters" – using networked commodity computers to create a scalable, distributed system.

Second generation

- The transition from the first generation, exemplified by Gmail and Google Search, to the second, represented by Facebook and Twitter.
- Gmail and Google Search operated largely in user-specific silos, with interactions relatively simple.
- In contrast, platforms like Facebook and Twitter, which introduced friend networks and followers, necessitated a more interconnected and dynamically updated database.

Challenges with Facebook

- Unlike Gmail's isolated data silos, Facebook required constant data migration and replication due to its interactivity.
- For instance, when a user likes a post, this "like" has to be reflected in real-time across various servers catering to different friends.
- This brings up a myriad of questions and challenges, especially when factoring in privacy settings and friend connections.

Eventual Consistency

- Facebook is an example of an "eventual consistency" database.
- This means that while a user might like a post, it might take some time for this action to reflect across all relevant servers.
- Such databases prioritize overall system health over real-time accuracy.

Engineer's Dilemma

- The engineers often overgeneralize solutions.
- Once they've crafted a solution for a specific problem, they might try to adapt it for broader applications, leading to the creation of new database forms.
- The engineer, when asked to pass the salt, begins developing a system to pass any condiment, believing that it'll be more efficient in the future.
- While the engineer's intention might be to create a versatile solution, it's an overcomplicated response to a simple request.

Emergence of New Database Forms

- The success of second-generation cloud companies led to the development of **BASE-style databases**, moving away from traditional ACID databases.

PART 02

BASE-style

BASE-style databases

- **Basically Available:** The system ensures availability, even in the presence of multiple failures.
- **Soft state:** The state of the system may change over time, even without input. This reflects the inherent uncertainty and inconsistency in distributed systems.
- **Eventual Consistency:** The system will become consistent over time, given that the system doesn't receive input during that time.

BASE-style vs ACID-style

- **ACID** (Atomicity, Consistency, Isolation, Durability) databases, like Oracle, Postgres, **MySQL**, **SQLite**, and **SQLServer**, ensure that data remains consistent over time.
    - They operate on the principle that data, once written, remains the same for all readers at any given moment.
- **BASE** (Basically Available, Soft state, Eventually consistent) databases, like **Mongo**, **Cassandra**, and **Google's BigTable**, operate on eventual consistency.
    - Data might be inconsistent across nodes for a while, but it will eventually synchronize.

BASE-style databases characterized

- Wide distribution without central locks.
- Use of fast networks, low-memory CPUs, and a multitude of disk drives.
- Data sharding with indexes for data location.
- Preference for documents over rows and columns.
- Flexible schemas allowing on-the-fly addition of key-value pairs.
- Emphasis on schema-on-read rather than rigid early schemas.

JSON's Role

- **JSON** became a crucial format, offering simple representation of key-value pairs.
- Provides fast parsing across various languages.
- Highly compressible, making it attractive for databases aiming for memory efficiency.

JSON example

```
{
    "user": {
        "id": 1,
        "name": "John Doe",
        "email": "john.doe@example.com"
    }
}
```

- This JSON structure represents a user with an id, name, and email.

Emergence of NoSQL Databases

- **CouchDB:** Aims for a cluster of unreliable commodity hardware.
- **MongoDB:** Popularized with the Node ecosystem, using JSON storage.
- **Cassandra:** Developed by Facebook engineers based on their experiences.
- **Elasticsearch:** Evolved from an effort to replicate Google search, focusing more on indexing than being a typical database.

Software as a Service (SaaS) Trend

- Offerings like Amazon's DynamoDB, Google's BigTable, and Microsoft's Azure allowed companies to use databases without heavy investments in in-house expertise or hardware.

Adoption by Major Players

- **Amazon:** uses Amazon Redshift.
- **Facebook:** uses Cassandra and RocksDB.
- **Twitter:** migrated from MySQL to Manhattan, its real-time, multi-tenant distributed database.
- **Apple:** Apple replaced some of its use of Cassandra with its in-house designed FoundationDB.
- **Netflix:** heavily uses Cassandra for its scalability and distributed architecture.
- **LinkedIn:** uses kafka and Espresso.
- **Uber:** Uber uses a mix of databases, including MySQL, Postgres, and Cassandra,
- **Microsoft:** uses Azure Cosmos DB.

PART 03

NoSQL

NoSQL

- NoSQL (often interpreted as "not only SQL") databases are non-relational databases that provide a way to store and retrieve data differently than traditional relational databases.
- The term "NoSQL" is a bit of a misnomer since many of these databases do use some form of SQL-like query language.
- The primary distinction is their non-relational architecture.
- There are several types of NoSQL databases: Document-based, Column-based databases, Key-value stores, and Graph databases.