Introduction to Megastore

Learn what Megastore is and why it was created.

We'll cover the following

Motivation

As desktop programs migrate to the cloud, interactive online services challenge the storage market to fulfill new needs. E-mails, shared reports, and social networking are developing at an enormous speed, pushing the limits of existing infrastructure. Handling the storage needs of these services is difficult due to the following demands:

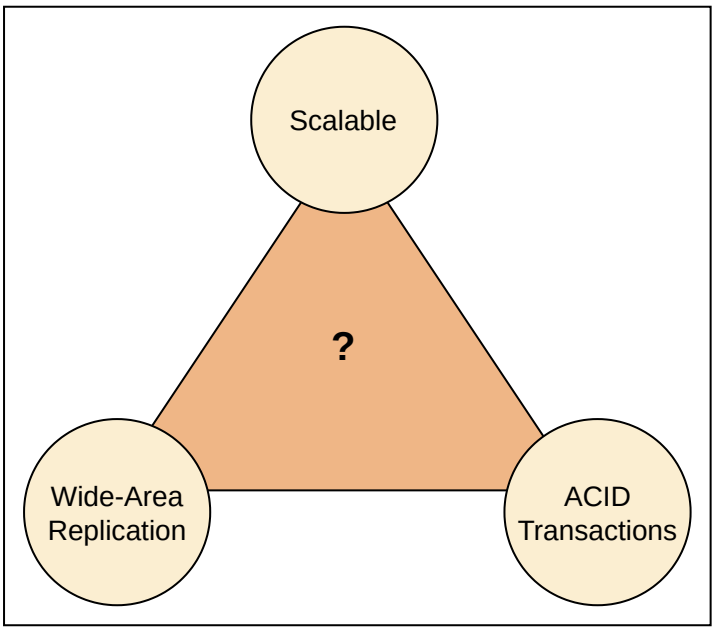
1. Applications must be extremely scalable due to the vast audience of potential consumers that the Internet brings. Using traditional databases such as MySQL since the datastore allows a service to be quickly developed, but expanding the service to millions of people demands a total overhaul of the storage infrastructure.
2. Organizations have to compete for users. This demands speedy product development and time-to-market. Usually, a NoSQL solution with some custom consistency models has its learning curve, and developers might be forcing the solution to the problem that is more amenable to traditional relational databases.

3. Low latency is essential for storage systems.
4. The application should give the user a consistent data view. The outcome of a change should be seen instantly and indefinitely.
5. The services should be highly available. The system should operate uninterrupted despite server or component failures.

Technology options

The demands above have trade-offs. Relational databases offer comprehensive capabilities for easily implementing applications, but scaling to hundreds of millions of people is tough. Although NoSQL datastores such as Google's Bigtable are very scalable, their restricted API and weak consistency models make application development more difficult. Transactions in Bigtable are possible at individual keys. For transactions across many keys, applications would need to explicitly use different mechanisms. Doing so makes the code complicated to write and manage. It is difficult to replicate data across distant data centers while maintaining low latency. It is even more difficult to ensure a consistent view of replicated data, particularly during breakdowns.

Hence, finding a globally scalable system that allows ACID (atomicity, consistency, isolation, and durability) transactions is hard.



There is no such globally scalable system that allows ACID transactions

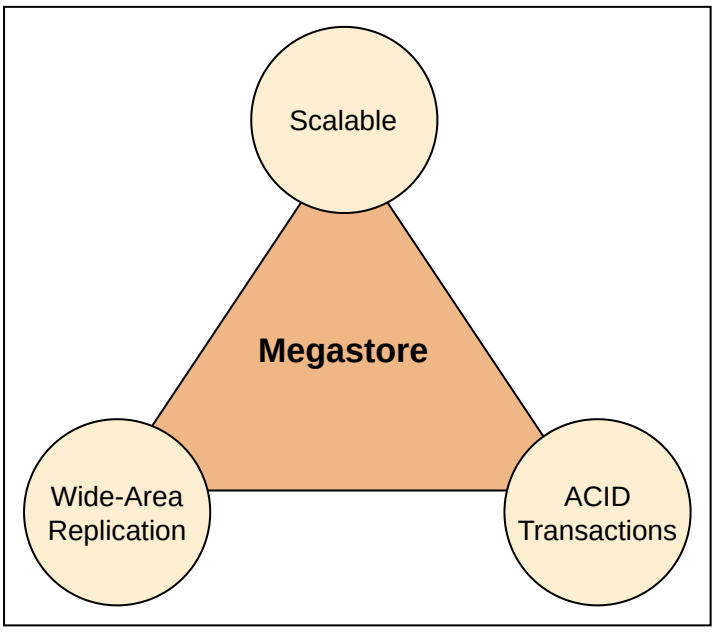
Megastore

To fulfill the storage needs of modern interactive web services, Megastore was developed. It provides higher availability and consistency by combining NoSQL's scalability with the ease of RDBMS. In a nutshell, Megastore provides completely serializable ACID semantics over fine-grained data partitions. Partitioning this way provides a seamless failover across data centers and synchronously replicates each write over a wide area network with acceptable latency.

Megastore adopts the middle road in the RDBMS vs. NoSQL technology field, partitioning the datastore and replicating each partition individually, offering complete ACID semantics inside partitions but only limited consistency guarantees between them. It includes standard database functionality such as secondary indexes. However, Megastore supports only those functionalities that could scale within user-tolerable latency limitations and according to the semantics that the partitioning system can support.

To manage more than three billion write operations per day and 20 billion read operations per day while storing a petabyte of data, Google has used Megastore internally on more than 100 commercial apps for several years.

Megastore is our solution for the SQL-based system that is highly scalable on a global level.



Requirements

Let’s list the requirements for designing a distributed storage system for managing interactive online services.

> **Note:** Since Megastore is built on top of Bigtable, it can provide what Bigtable does, but Megastore goes beyond that.

Functional requirements

The functional requirements of such a system are as follows:

- **ACID transactions:** The system should facilitate interactive applications by offering fully serializable ACID semantics over far-off replicas with reasonable latency.

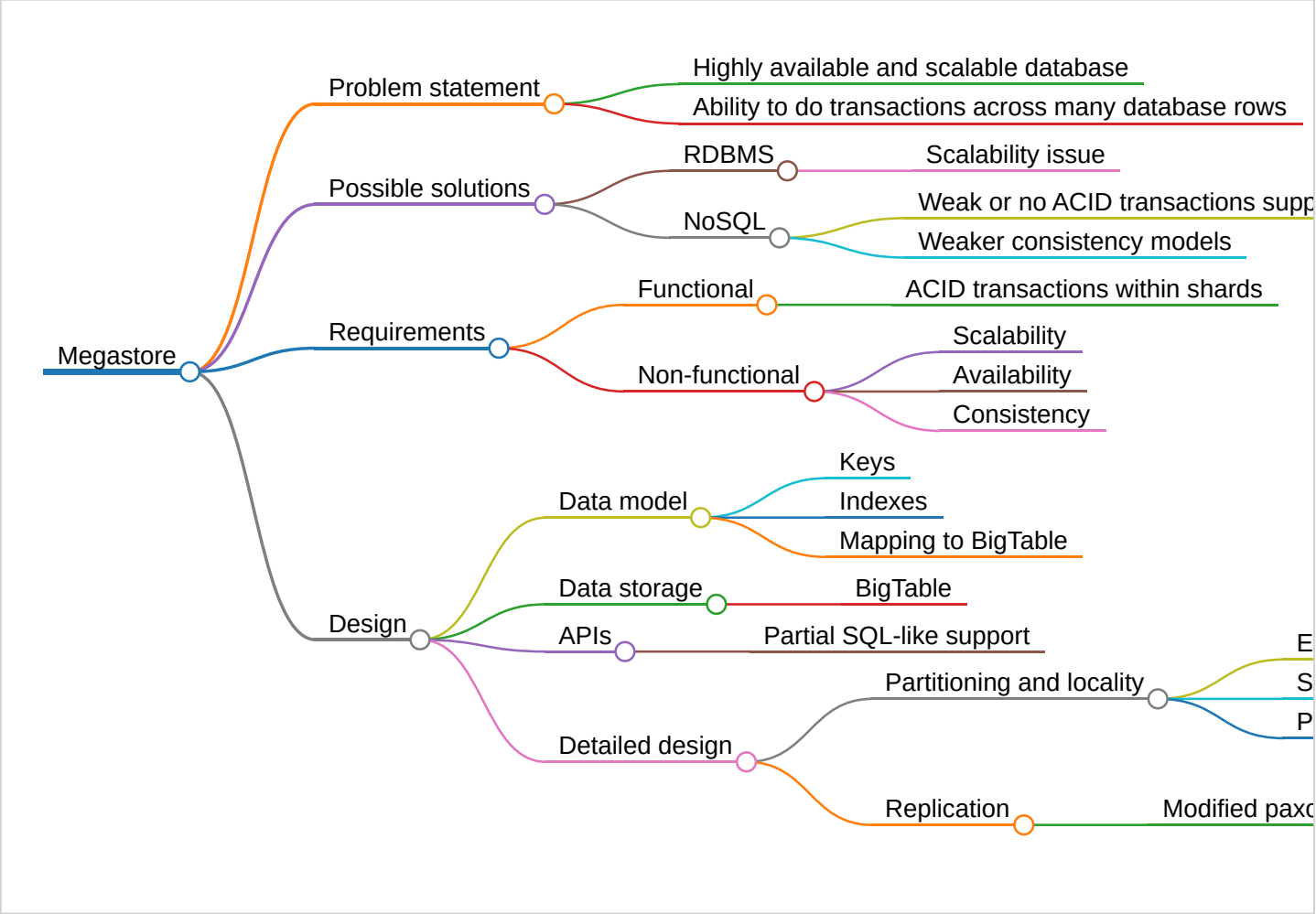
Non-functional requirements

The non-functional requirements are as follows:

- **Scalability:** The system should scale up to petabytes of primary data across data centers.
- **Availability:** The system should be highly available. The system should operate uninterrupted despite server or component failures.
- **Consistency:** The system should be consistent, and the users should be able to see a consistent state of data within a shard.

Bird's eye view

In the next lessons, we will design and evaluate Megastore. The following concept map is a quick summary of the problem Megastore solves and its novelties.



> **History of Megastore:** Developer frustrations with key-value-based stores such as Bigtable prompted Google to shift towards a relational database-like architecture—something closer to the traditional relational data model. Google's Bigtable, the most well-known key-value store, is still in widespread usage today at Google. A relational schema system, cross-row transactions, consistent replication, and a strong query language were necessities for developing many OLTP (online transaction processing) applications that were either missing in Bigtable or for applications that had to circumvent or implement them at the application level. An initial solution to these issues included constructing transaction processing systems like Megastore on top of Bigtable. In the next lesson, we will discuss how Megastore ensures availability and scalability.

Back
Quiz on Bigtable

Next
High-level Design for ...

Mark as Completed