Lecture #17: Spanner and the evolution of its Storage Engine

15-721 Advanced Database Systems (Fall 2024)
https://www.cs.cmu.edu/~15721-f24//
Carnegie Mellon University
Author: Luojia Hu

1 Introduction

Spanner is Google's globally distributed database management system that, fundamentally, a SQL database system with global scale and strong consistency. It currently stores 15 exabytes of data and handles 5 million queries per second. It powers critical Google services we probably use every day, including Gmail, YouTube, and Google cloud services. This lecture explores the core architectural components, scalability features, and innovative technologies that make Spanner unique, with a particular focus on its geographic replication strategy, scalability mechanisms, TrueTime implementation, and storage engine evolution.

2 Spanner Architecture

2.1 Geographic replicas

Figure 1: Example of Spanner system relicas

Spanner is geographically replicated at planet scale. Figure 1 is an example of a real database. The system has 3 replicas in the United States, and 2 in Europe. The 3 replicas in the US result in a **quorum**, where transactions can complete relatively quickly within that local quorum.

Spanner is a classic, horizontally scalable system, so we can run anything from a fraction of a server to thousands of servers on each replica. Spanner stores about 15 exabytes of serving data and is running about 5 billion queries per second.

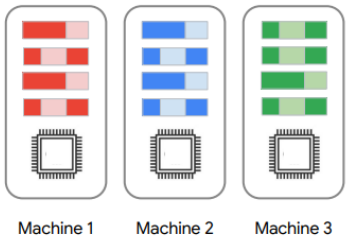
2.2 Scalability

Spanner has high scalability. To explain this, consider the example database with a table called `Songs`, holding information about the audio on an album. To achieve load balancing, Spanner needs to take key ranges and distribute the rows over machines. If one machine is frequently accessed, Spanner will automatically split up that data and assign it somewhere else to do dynamic load balancing.

```
CREATE TABLE Songs(
    ArtistID    INT64 NOT NULL,
    AlbumID     INT64 NOT NULL,
    TrackID     INT64 NOT NULL,
    SongName    STRING(MAX),
    ...
) PRIMARY KEY (ArtistID, AlbumID, TrackID)
INTERLEAVE IN PARENT Albums;
```

Another key thing in Spanner is **interleaving**. Interleaving is a mechanism of co-locating rows from multiple tables sharing a prefix of primary keys.

In the above example of table Songs, we notice that an Artist can have multiple Albums, and an Album can have multiple Tracks and SongNames. This constitutes a parent-child relationship. The child table is co-located with the parent table, and it can be interleaved in the parent, so that all child rows whose primary key is prefixed with key parts are stored physically next to the parent row, like what figure 2 shows.



ArtistId	AlbumId	AlbumName	TrackId	SongName
1	1	Best of Dizzy Gillespie		
1	1		1	A Night in Tunisia
1	2	Afro-Cuban Moods		
1	2		1	Exuberante
3	1	Milestones		
3	1		1	Straight, No Chaser
3	2	All Blues		
3	2		1	All Blues
3	2		2	Flamenco Sketches
3	2		3	So What
5	1	Psy 6 (Six Rules)		
5	1		1	Gangnam Style

Figure 2: An example of Interleaving. Here multiple entries sharing the same prefix (ArtistId, AlbumId) will have their prefix stored only once.

This technique is a non-standard SQL, but it lets Spanner store data in a pre-joined format, and it's highly frequent for there to be locality in the system. It's fundamental to the scalability of Spanner system.

2.3 TrueTime

Spanner assigns timestamp to every operation. **Truetime** is Google's innovative solution for maintaining consistent global timestamps across their distributed database system. It provides guaranteed bounds on clock uncertainty, enabling strong consistency in distributed transactions.

The core components behind Truetime include two independent reference time sources: one is the GPS receivers for satallite-based timing, and the other is the atomic clocks for backup and redundancy. Corre-

spondingly, each data center has multiple time masters, including GPS time masters with satellite receivers, and armageddon masters with atomic clocks.

In Truetime, time is represented as an interval with bounded uncertainly. The API provides three core methods:

- `TT.now()`: Returns an interval called $[earliest, latest]$. The actual time is guaranteed to be within the returned interval.
- `TT.after(t)`: Returns True if t has definitely passed
- `TT.before(t)`: Returns True if t definitely hasn't arrived.

The uncertainty ϵ of `TT.now()` calls typically ranges between 1-7ms in practice. This is achieved through regular synchronization with time masters every 30 seconds, algorithms to detect and reject incorrect time sources, and careful monitoring of local clock drift between synchronizations.

Spanner uses Truetime to establish transaction ordering by:

- Waiting for the uncertainty interval to pass before committing transactions.
- Using timestamp ranges to guarantee consistent reads across globally distributed machines.
- Ensuring that if transaction T1 commits before T2 starts, T1 receives an earlier timestamp.

In this way, Spanner can provide externally consistent transactions and lock-free reads at a global scale.

2.4 Storage Engine

Underneath all of the Spanner applications, the storage engine is responsible for most of the system's single node operation.

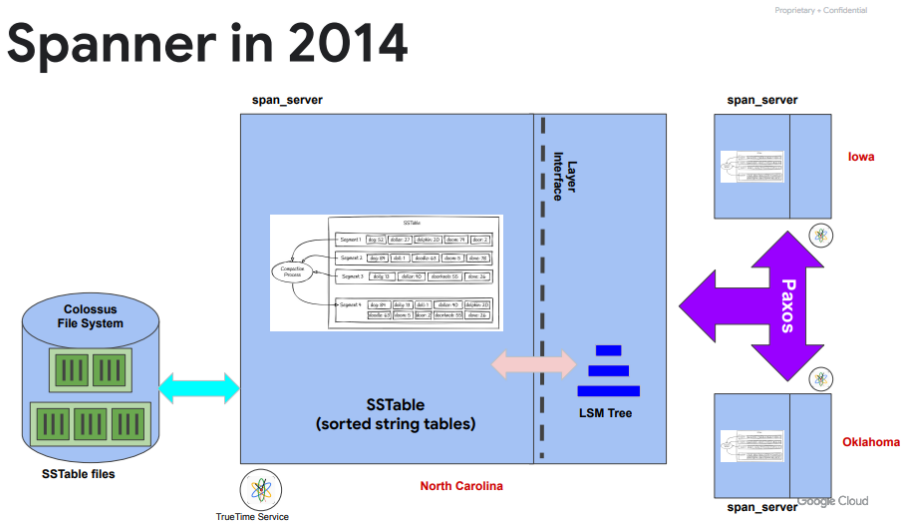


Figure 3: Previous Storage Engine

Previously, Google's Spanner system looked as shown in figure 3. It was a fork of the Bigtable code base, and used sorted string tables to store unstructured data. The downside is that everything was self-describing and enormously redundant. Compression is needed to reduce the redundancy.

With Ressi (shown in figure 4), the new storage engine, almost everything below the layer interface is replaced, including the file format, IO management, block cache, access methods, mem tables (write buffers), and so on. This project started in 2013 and was completed in early 2022, much longer than the initial 2-3 years estimate. Ressi packs layout with type-dependent encoding, run length coding, delta coding, compression, and entropy coding.

The migration to Ressi had many challenges, including:

- Live migration requires maintaining the availability while doing the switch.

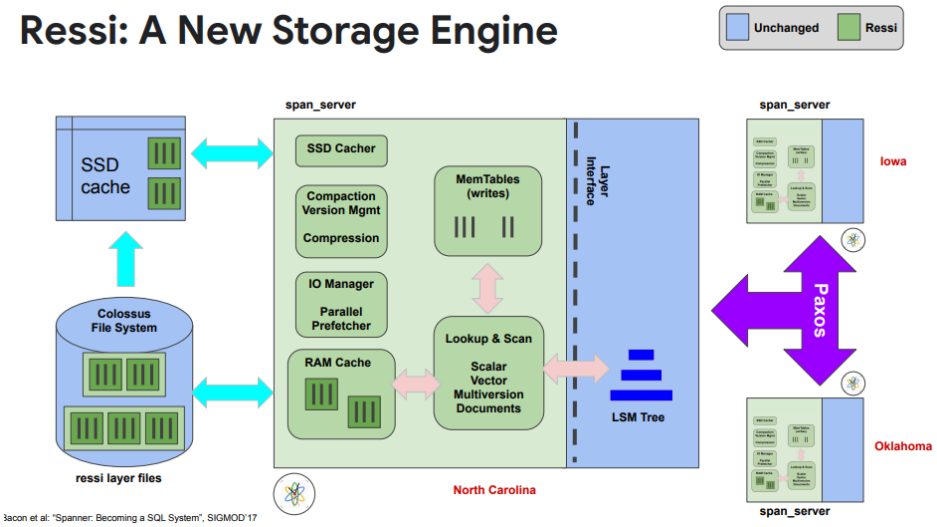


Figure 4: Ressi, Google's new underlying storage engine

- Since the system is critical for billions of users, performance and scale should not be obviously affected.
- Privacy needs to be maintained, but cannot directly inspect data for debugging, so privacy-preserving experimentation methods are needed.

To achieve live migration, an experimental framework was built, which operates on a single tablet with a small portion of the database. It is a place to test changes safely without affecting production. The data are stored in RAM or encrypted on disk.

3 Conclusion

Google Spanner demonstrates how modern distributed database systems can achieve seemingly contradictory goals: global scale with strong consistency, high availability with disaster recovery, and automatic sharding with transactional integrity. The data is horizontally shared across servers located in different regions of the world. Spanner provides external consistency, which is stronger than linearizability and serializability. It uses TrueTime for global timestamp coordination. It uses Ressi as the storage engine. Through the above innovative features, Spanner has set new standards for what's possible in distributed databases. The system's evolution from its initial implementation to its current state, managing 15 exabytes of data and processing 5 billion queries per second, showcases both the technical challenges and solutions in building planet-scale database systems. As distributed systems continue to evolve, Spanner's architecture and design principles provide valuable lessons for the future of database technology.