Spanner : Google's Globally-Distributed Database

CS 237, Spring 2020
Presented by Rob Gevorkyan

Agenda

- What is Spanner?
- Data Storage Architecture
- Data Model and Querying Language
- TrueTime
- Implementing Transactions

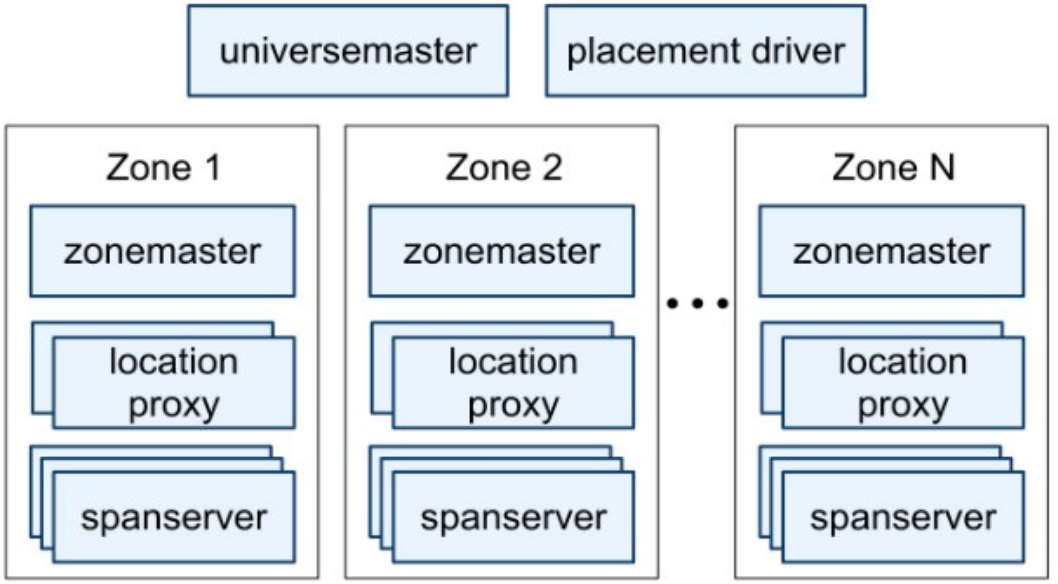
What is Spanner?

- Spanner is a **scalable**, **multi-version**, **globally-distributed**, **synchronously-replicated** database
  - **Scalable**: Data centers and servers can be added or removed as needed for availability and performance
  - **Multi-version**: Data has a timestamp, enabling historical data queries
  - **Globally-Distributed**: Data centers can be separated by great distances
  - **Synchronously-replicated**: Replication results in multiple up-to-date copies of data

What is Spanner?

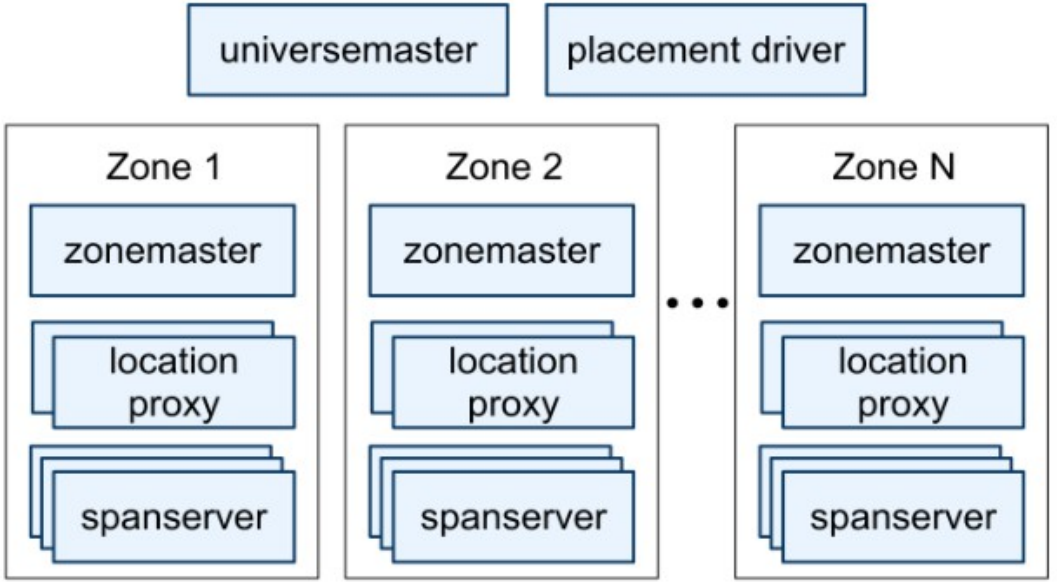
- A database that guarantees **external consistency** and **global consistency**
  - **External consistency:** Transactions are processed in a way equivalent to a single machine processing all transactions. Analogous to distributed shared memory, which provides an abstraction for many memories as one centralized memory
  - **Global consistency:** Commits that are made are visible to all replicas. Queries made from different data centers give the same values.

Data Storage Architecture



- Universe: The entire Spanner deployment
- Zone: A cluster of machines in a data center used to serve data to geographically close clients
  - A universe may have an arbitrary number of zones.

Data Storage Architecture

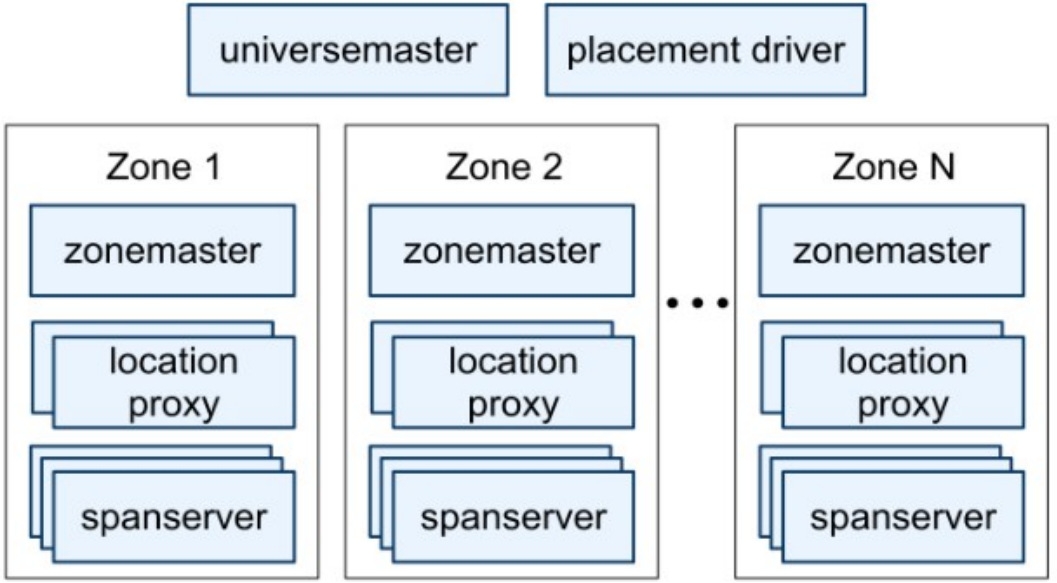


- universemaster: provides status information about all zones in universe for debugging
- placement driver: handles movement of data between zones

Implementing Transactions

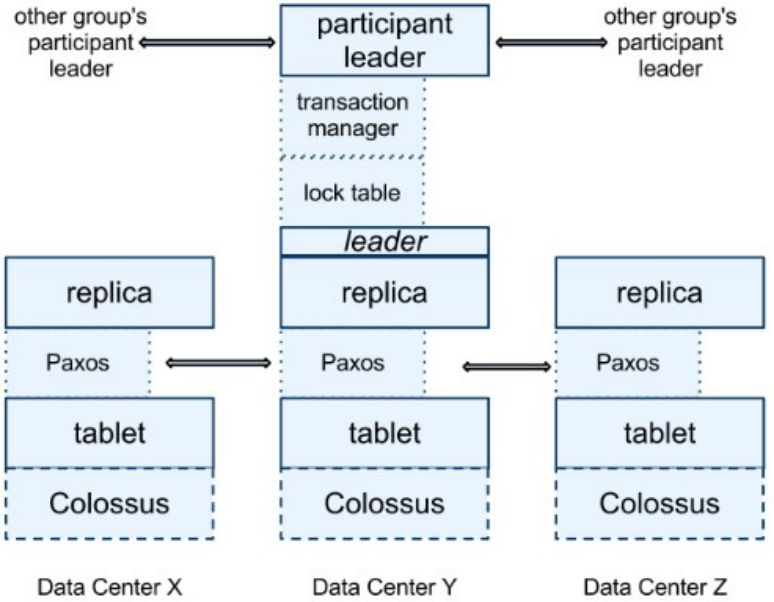
- Read-Only Transactions
  - The scope of a query is the set of keys needed to serve it.
    - If the scope is confined to a single Paxos group, the client issues the transaction to that group's leader. This does not require coorddination.
    - If the scope spans multiple groups, each of the Paxos groups must coordinate using the Paxos protocol to agree upon the result of the query.

Data Storage Architecture



- spanserver: stores data
  - A zone may have hundreds to thousands of spanservers.
- zonemaster: assigns data to spanservers
  - A zone has exactly one zonemaster

Data Storage Architecture : Spanserver



- Colossus: Distributed file system
- Tablet: A segment of a larger table. A group of tablets together form the data in a single
- Paxos state machine: participants in Paxo protocol for consistency

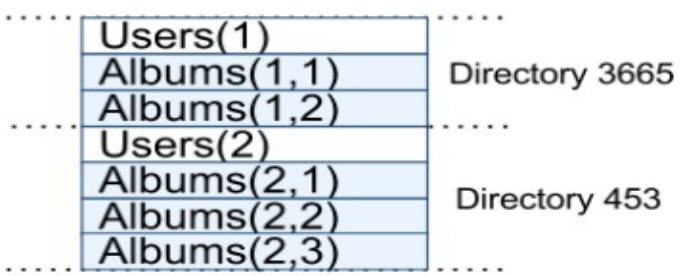
Data Model and Query Language

- Spanner provides a semi-relational interface with a query language similar to SQL, with some additional features
  - Database: Split among many zones geographically separated
  - Tables : Split among many tablets replicated across data centers
  - Rows: Collection of values for a variety of data columns, timestamped.
  - Columns: Values from the internal key-value store.

Data Model and Query Language

```
CREATE TABLE Users {
  uid INT64 NOT NULL, email STRING
} PRIMARY KEY (uid), DIRECTORY;

CREATE TABLE Albums {
  uid INT64 NOT NULL, aid INT64 NOT NULL,
  name STRING
} PRIMARY KEY (uid, aid),
  INTERLEAVE IN PARENT Users ON DELETE CASCADE;
```



- Interleave in: Declares a relationship that makes values between two tables stored geographically in the same place.
  - Directory: The smallest unit of replication. i.e. rows cannot be replicated. Only directories can.

TrueTime

Method	Returns
$TT.now()$	$TTinterval$: $[earliest, latest]$
$TT.after(t)$	true if t has definitely passed
$TT.before(t)$	true if t has definitely not arrived

- Fulfills the need for global absolute time, with bounded uncertainty
- Implemented with
  - GPS
  - Atomic clocks
- Uses a variant of Marzullo's algorithm, an approach for estimating time from a number of possibly noisy sources
- The average error bound is 4ms

Implementing Transactions

- Read-Write Transactions
  - Writes occuring in transaction are buffered at client until commit. Reads therefore do not see the effect of the writes.
  - Reads use wound-wait to avoid deadlocks.
  - Utilizes two-phase commit.
  - Timestamps provided by TrueTime have an important role in ensuring consistency.

Implementing Transactions

- Schema-Change Transactions
  - Support provided for atomic scheme changes **in a single datacenter**.
  - Any transactions occurring with a TrueTime timestamp before the timestamp of a scheme change can proceed unblocked. Any occurring afterwards must block until after the schema change has occurred.