Spanner

Stephanie New

Overview

Scalable, multi-version, globally distributed, and synchronously replicated database.

Supports non-blocking reads in the past, lock-free read-only transactions, and atomic schema changes.

To support externally consistent distributed transactions at a global scale, it uses the TrueTime API that exposes clock uncertainty.

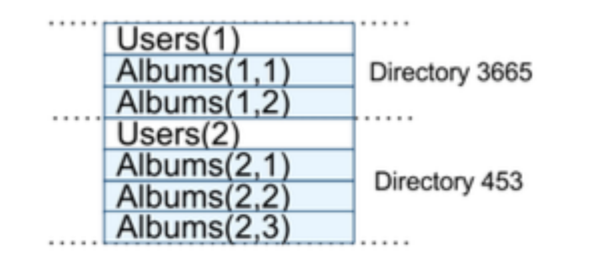
Motivation

Popularity of Megastore over Bigtable because of its semi-relational data model and synchronous replication, despite its poor write throughput.

Spanner evolved from a Bigtable-like versioned key-value store into a temporal multi-version database.

Data is stored in semi-relational tables, and Spanner provides a SQL-based query language and supports general-purpose long-lived transactions.

Data Model



An application using Spanner creates one or more databases in a Spanner deployment. Each database can contain an unlimited number of schematized tables.

Not purely relational because every table is required to have an ordered set of one or more primary-key columns.

Every Spanner database must be partitioned by clients into one or more hierarchies of tables.

As a Distributed Database

Data is versioned, and each version of data is automatically timestamped with its commit time by the TrueTime API.

External consistency: If a transaction T1 commits before another transaction T2 starts, then T1's commit timestamp is smaller than T2's.

Using TrueTime, Spanner is able to assign globally-meaningful commit timestamps to transactions, which reflect serialization order.

TrueTime API

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

TT.now() is guaranteed to include the absolute time within the interval.

There are two forms of time reference, GPS and atomic clocks, because they have different modes of failure.

Implemented by a set of time master machines per datacenter and a time slave daemon per machine. Every daemon polls a number of masters to reduce vulnerability from any one master.

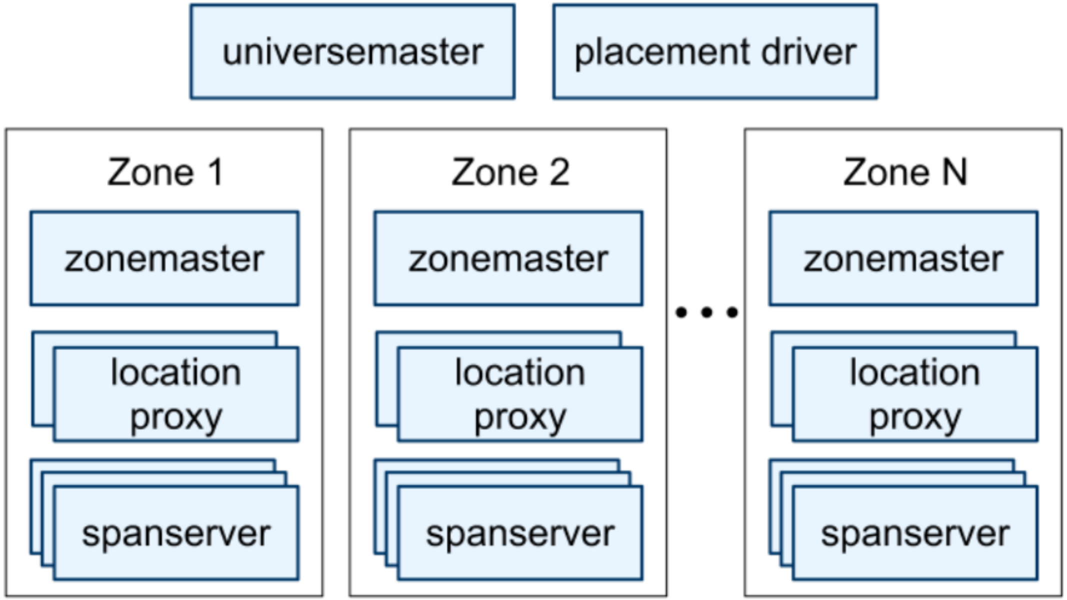
How to Keep Uncertainty Small?

Define the instantaneous error bound as ϵ, which is half of TTinterval's width.

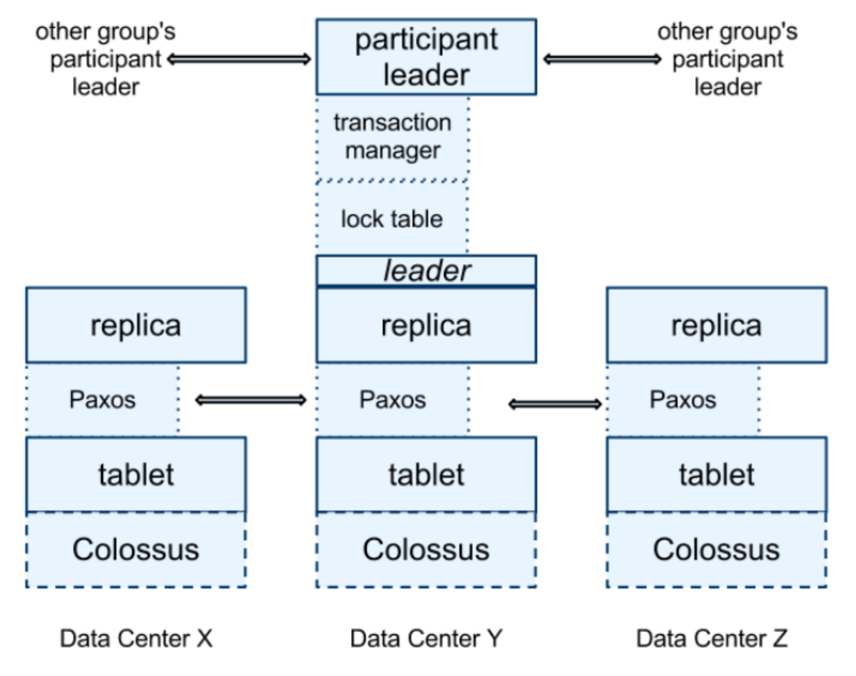
ϵ is derived from conservatively applied worst case clock drift. ϵ also depends on time-master uncertainty and communication delay to time masters.

In Google's production environment, ϵ is typically a sawtooth function over time, varying from about 1 to 7 ms over each poll interval. Thus, the average is 4ms most of the time.

Spanner Deployment: Universe



Spanserver Stack



(key:string, timestamp:int64) -> string

Concurrency Control

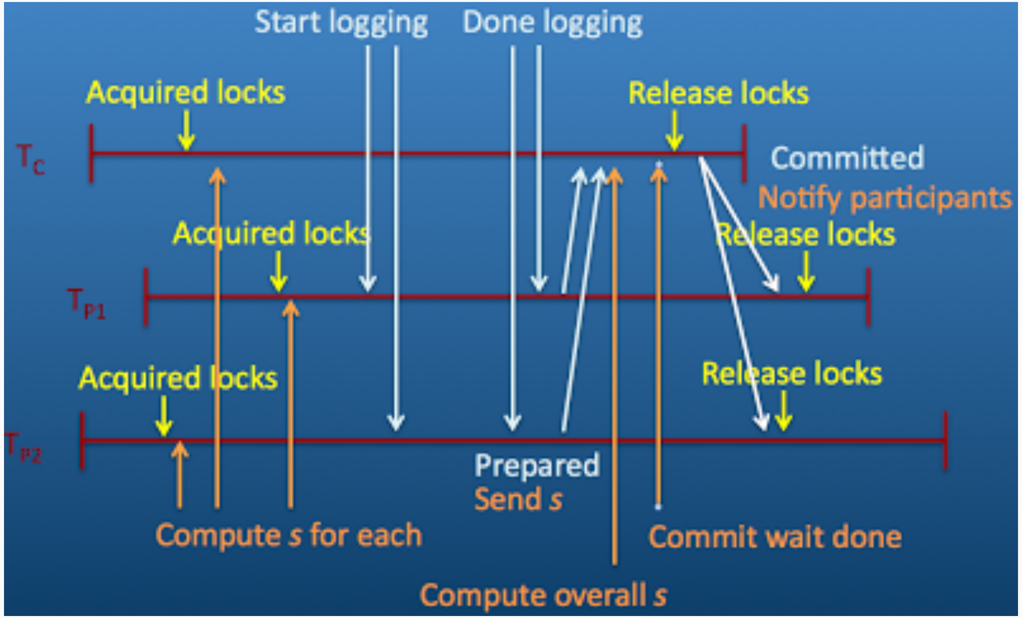
Spanner supports read-write transactions, read-only transactions, and snapshot reads.

Standalone writes are implemented as read-write transactions.

Non-snapshot standalone reads are implemented as read-only transactions.

A snapshot read is a read in the past that executes without locking.

Read-Write Transactions



Read-Only Transactions

A read-only transaction executes in 2 phases:

1) assign a timestamp s_{read} .

2) execute the transaction's reads as snapshot reads at s_{read}.

Snapshot reads can execute at any replica that is sufficiently up to date (i.e. s_{read} is less than or equal to a t_{safe}).

Why TrueTime?

Lock-free reads can be implemented without TrueTime using only sequence numbers because read-only transactions are also serialized by coordinating leaders and Paxos groups.

TrueTime benefits snapshot reads, reads in the past, the most. By giving a time in the past, the snapshot read can get a consistent read of all variables requested at that given time.