Evaluation of Spanner

Evaluate if Spanner fulfills the non-functional requirements.

We'll cover the following

Consistency

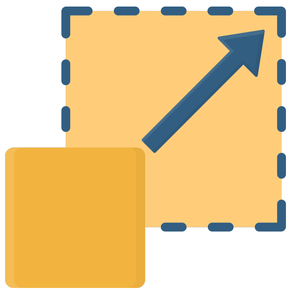
Spanner provides external consistency (the strongest kind of consistency) via the TrueTime API. Regardless of whatever replica performs the read, a strong read in Spanner will always see the results of all transactions committed before the operation began. Spanner relies on TrueTime to generate timestamps from a decentralized, virtual version of the world clock.

When a transaction is committed in Spanner, a timestamp is added to it from a trusted source, TrueTime. This ensures that the commit sequence of transactions can be linearized. There is a strict order to execute all transactions to maintain external consistency.

Whenever a client observes a transaction T2 beginning to commit after a previous transaction T1 has completed, the system will give T2 a timestamp that is later than T1's. With TrueTime, clients produce timestamps that increase monotonically. Let’s say an app executes two transactions, $T1$ and $T2$. If $T2$ starts after $T1$ has ended, the timestamp assigned to $T2$ will be greater than the timestamp of $T1$.

Spanner takes advantage of TrueTime's timestamping functionality to precisely date and time its events, such as transaction commit times. In particular, Spanner provides a timestamp for each transaction that accurately reflects the moment the network processes the transaction. With ordering guarantee and multi-version concurrency control of Spanner on timestamps, the writes are not blocked and clients can conduct consistent reads across an entire database, regardless of where it resides in the cloud.

Scalability



Spanner separates computing resources from data storage, allowing the pool of processing resources to scale, descale, or reallocate independently of the underlying storage. Spanner relies on Colossus for the scalability of data. Moreover, it uses `movedir` to manage the movement of data which distributes the load and utilizes the newly added capacity.

Availability

Spanner provides up to 99.999% availability due to TrueTime clocks and synchronous replication. With Spanner, data is replicated so that it is always accessible. All the information in Spanner is structured in rows at the highest level. Spanner creates replicas that are copies of these rows and saves those replicas in more than one location.

Spanner employs a Paxos-based, synchronous replication strategy where all write requests are subject to a vote by all voting replicas before being committed. Due to the globally synchronized nature of Spanner's replication, users can access the most recent database version from any read-write or read-only replica. Each Paxos group relies on a majority quorum. Thisfa implies that as long as a majority quorum can be established, the failure of a few nodes will not impact availability. For across Paxos group operations, Spanner uses two-phase locking (2PL) and two-phase commit (2PC).

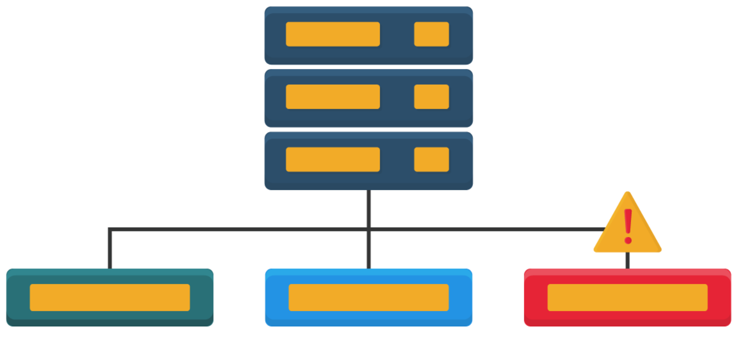
Traditionally, such protocols (2PL and 2PC) are considered slow, especially when participants can fail. Though, in Spanner, each participant in 2PL and 2PC is a member of a Paxos group. Progression in 2PL and 2PC is recorded in the underlying Paxos logs. That means, even if a participant fails during 2PC, the new member of the Paxos group can resume where the old one left off. Such a mechanism enables Spanner to have higher performance and higher availability.

Once a database is split, Spanner makes a copy of each part. A split stores a group of adjacent rows sorted with their primary key. Spanner serves each replica from its separate failure zone, yet the partitioned data is physically kept together in the replica. Splits can be stored and replicated using Paxos. One Paxos replica in each group takes the role of the leader. Any replica (read-write or read-only) can serve reads without communicating with the leader.

Moreover, with Spanner, data can be distributed across multiple regions and continents, bringing it closer to the users and services that require it.

Fault tolerance

Spanner is fault tolerant by ensuring the replication of data. Spanner allows users to replicate the data in multiple regions. Therefore, even if data in a single region is lost, other regions have the required data. The system will create a new instance and copy the data from other regions.



Conclusion

The Google Spanner database is a system that provides the highest level of consistency guarantees, including linearizable reads and writes, an atomic commit, and transactions with serializable isolation. Spanner can do all this while being highly scalable, supporting high amounts of data and transaction throughput, and enabling global data distribution.

Spanner uses traditional methods. It utilizes the Paxos for the replication of state machines, two-phase locking for serializable isolation of transactions, and two-phase commit for atomic commit. However, these well-established options are very predictable from an architectural perspective, even though operating them in practice takes great engineering effort. However, the use of GPS and atomic clocks by Spanner makes it unique. Considering uncertainty to occur within the execution of the transaction, TrueTime utilizes high-precision clocks to provide physical time's upper bounds and lower bounds. Spanner ensures that timestamps are causality-consistent by waiting out the uncertainty and interval.

To summarize, Google Spanner is a globally distributed, ACID-compliant, replicated, auto-sharded, and transaction processing database. Spanner is built on Google's specialized network and has been field-tested by Google's billion-user services. It has an uptime of 99.999% and requires no downtime for upgrades or changes to the database structure.

System design wisdom in Spanner

- Spanner's major innovation of TrueTime is an example of challenging conventional wisdom. While it is well understood that perfectly synchronizing different clocks in a large distributed system is almost impossible, Spanner asks what the next best thing is. Spanner's answer is that we can put a bound on the clocks' drift which we can sustain for extended periods. This realization unlocks many benefits. Spanner then shows how we can build a strongly consistent distributed database with good performance and get nice properties, such as linearization and serialization.

- To provide lower bounds on clock drifts, the TrueTime API heavily relies on Google's private network (that in itself provides nice properties in terms of controllable latency, throughput, and availability) and a carefully designed network of GPS and atomic clocks. It is an example of how a good system (a system that is working as per its service-level agreement (SLA) for extended periods and provides value to its clients) relies on other good subsystems.

In summary, Spanner moved the state-of-the-art for distributed databases using the fascinating innovation of TrueTime and other abstractions on top of it.

← Back
Database Operations i...

Next →
Quiz on Spanner

☐ Mark as Completed