Heather Linich

December 1, 2014

Spanner: Google’s Globally-Distributed Database

OSDI 2012

A paper at the 2012 USEXIX conference introduced one of Google’s new products: Spanner. Spanner is a new database that sets itself apart due to its ability to be scalable and globally distributed. Spanner functions by creating horizontal partitions (also known as shards) in the database across many sets of state machines. As the quantity and volume of data or servers change, Spanner will automatically reshard and relocate data if there are failures and to balance the data load. One of the big draws of Spanner is its ability to handle millions of machines if needed.

Spanner is meant to be a replacement for Google’s BigTable data storage system. Many BigTable clients were unhappy with the lack of capability of the program to handle transactions. Functionality for handle distributed transactions is one of the main components of Spanner.

One of Spanner’s main features is administering replicated data across multiple datacenters. Managing configurations for data duplication is able to be dynamically regulated by applications. Applications can then stipulate any necessary restrictions on the data that each datacenter houses and manages. Data can be transferred between datacenters at any time in order to achieve balance in system resources.

For Spanner’s implementation, directory abstraction is used. Directory abstraction mainly controls geographic locality and replication. The Spanner ‘universe’ is structured as a set of zones which define the set of datacenter locations throughout which data can be replicated. In the event of the establishment of new datacenters or the shutdown of datacenters, new zones can be created or deleted accordingly.

Spanner timestamps its data which helps it to function like a multi-version database. Spanner logs each Paxos (Paxos is a protocol for achieving a consensus result from a group of unreliable processor participants - *Wikipedia*) two times and the implementation occurs in a pipeline so as to help throughput in the event of any WAN latencies.

Spanner implements directories which are essentially a bucketing abstraction that has a set of keys that typically share a prefix. This enables the application to carefully manage the locality of data by assigning specific keys. Each directory comprises information so the same replication configuration is applied to all data within that particular directory. A significant distinction of Spanner from BigTable is that Spanner functions as a container which may collect information on multiple partitions and directories that are often accessed together. If a directory eventually grows to a prohibitively large size, Spanner will shard it into as many fragments as needed to continue efficiently functioning.

Spanner implementation is able to support three types of transactions: read only, read-write and snapshot reads. For Spanner’s purposes, standalone writes are executed as read-write transactions and non-snapshot standalone reads are read-only. Read-only provides the performance benefits typically seen with snapshot isolation but must specifically state that there will be no writes. With read-only transactions, the reads execute without locking at a specific timestamp that is provided by the system. This is so that any writes that attempt to execute are not blocked. A snapshot read is a past read that does not lock upon execution and either has a specified timestamp or an upper bound on the timestamp’s age which will prompt Spanner to assign a timestamp as necessary.

Read and write transactions make use of two phase locking and can have timestamps assigned whenever all of the locks have been acquired but before any locks are released. Within a read-write transaction, reads will use a wound-wait in order to avoid deadlock. Writes are buffered until a commit takes place and therefore reads do not see the outcomes of writes.

Spanner was first experimentally introduced in 2011 during a reworking of a Google advertising product called F1. The team working on F1 chose to implement Spanner for several reasons. Spanner offers both automatic failover and synchronous replication. Previously, failover was challenging to manage and there was a significant possibility of downtime and/or data loss. Spanner also improved efficiency for the team by eliminating manual resharding. Finally, F1 necessitates strict transactional semantics which made it difficult to make use of noSQL systems. These semantics mandate transactions across indiscriminate data, consistent reads and secondary indexes on data. Spanner allowed F1 to meet each of these requirements.

Spanner’s timestamp procedures allowed F1 to effectively preserve in memory data structures that are compiled from the database state. F1 retains a log of all changes and updates which has been built into each transaction in Spanner. F1 creates records of data at a certain timestamp in order to initialize the data structures and then updates the structures with any necessary changes.

The Spanner and F1 teams have been working closely in order to move Google’s advertising backend processes from the existing MySQL to Spanner. Currently, the default setting for all program writes are sent from F1 to Spanner instead of MySQL. Spanner is presently working to advance the performance and functionality of its backup and restore system.

Spanner is a tool that can easily help businesses efficiently and easily organize their data. The ability of Spanner to be scalable is a huge draw of the program as a business would not need to completely change systems as they grow and if their data would exceed a typical data center, Spanner can easily accommodate. Spanner can provide services to a small start-up or a fortune 500 company.

Since Spanner uses data centers across many locations worldwide, users can rest assured that there will be no outages as resources can easily and quickly be rerouted to other data centers to keep everything up and running. This would be very reassuring to internal teams who are vetting out a company’s disaster recovery policies because the data would always be available and backed up.

Due to the large scope of the program, the versatile resources, reliability of the system and ease of scalability, the majority of companies in existence that need a database solution could easily implement Spanner.