SUMMARY:
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber.
Bigtable: a distributed storage system for structured data.
In *Proc. USENIX Symposium on Operating System Design and Implementation (OSDI'06)*, 2006.

DATE: 1 February 2010

Bigtable is a Google distributed storage system designed to manipulate large amounts of (structured) data with high availability, low latency and fault-tolerance goals. Unlike relational databases, Bigtable provides only a simple data model with emphasis on clients being in control of their data attributes (format, layout, locality etc.). A multi-dimensional sorted map is the fundamental data model. The map is indexed by a unique key made up of a row key, a column key, and a timestamp. Data is maintained by lexicographic order by (string based) row keys, allowing the clients to optimize data access and locality (by selecting appropriate row keys). The column keys are grouped into *column families*. Each column family stores similar type of data under a *family:qualifier* column key. Unlike a traditional relational database table, Bigtable can have an unbounded number of columns. Once the column family has been defined, new columns can be added by using unique qualifiers within the family. The row key and the column (family:qualifier) key uniquely identifies a data cell. Within each cell the data contents are further indexed by timestamps providing multiple versions of the data in time. Bigtable only allows for single row transactions.

A table is split into smaller manageable range of rows called *tablets*. Tablets allow for efficient data management and distribution operations. Tablets are assigned to distributed tablet servers governed by a single master server. Clients communicate directly with the tablet servers. Each tablet server stores the data in immutable *SSTable* files. New committed updates are first stored in a memory based *memtable*. Read operations are performed against a combined view (of SSTables and memtable). Periodically a memtable is flushed into an SSTable allowing for efficient memory utilization. Caching and (client specified) Bloom filters are utilized for efficient read operations. Bigtable system relies heavily on Chubby for master server selection. The tablet server persistent needs are fulfilled by the Google File System (GFS). Bigtable is most suitable for retrieving data based on primary keys rather then querying for complex data relationships.

SUMMARIZED BY: Atif Khan (a78khan)