**Bigtable: A Distributed Storage System for Structured Data**

**Introduction:** Bigtable is a distributed storage system designed to store structured data. Bigtable does not support a relational model but uses a data model. Data is indexed using row and column names that can be arbitrary strings.

**Data Model:** A sparsely distributed, persistent sorted multi-dimensional map, indexed by a row key, column key and timestamp. Each value in the map is uninterpreted array of bytes.

**Rows:** Follow a lexicographic order by row key, which are arbitrary strings. The row range in bigtable is dynamically partitioned. Each row range is called as tablets and is the reason for load balancing.

**Column Families**: Column keys are stored as families. This helps in the access control, disk as well as memory accounting.

**Timestamps:** 64 bit integers that help in picking up a unique record in a bigtable.

**API:** Big table API provides functions for creating and deleting tables and column families. Allows cells to be used as integer counters, permits single row transactions that helps perform read-write-modify transactions on any single row key.

**Building blocks:** Google SSTable file format is used to store bigtable data. Bigtable relies on highly available distributed lock service, chubby. Bigtable uses chubby for variety of tasks like, to ensure at most one master is active at any point of time, to store bigtable schema information, bootstrap the location of bigtable data.

**Implementation:** Bigtable components are Client library, master and tablet server.

Master takes care of assigning tablets to tablet servers, keeping track of tablet servers that are active. Tablet server takes care of splitting tablets that have grown large and read/writes that happen on the tablet.

**Tablet Location:** There is a three level hierarchy in storage of tablets. First is a storage file called root tablet present in chubby. This file has Metadata rows about all the tablets. It contains the location of row key, which is an encoding of tablet’s table identifier in end row.

Client library caches tablet locations. If it does not have the location/ store incorrect, it moves up the tablet location hierarchy recursively. Secondary information such as server start up logs and operation logs are also stored in the METADATA table to look for help in debugging and analytics.

**Tablet Assignments**: The Master tablet uses chubby to place a lock on one of the cells. The master tablet later scans for available cells, assign operations and keeps track of status of the operations by checking with the tablet servers. If any of the tablet server is dead, the master reassigns the operation and ensures availability of data is maintained without any trouble.

**Compactions:** As write happens, size of memTable increases. After a limit, the table size is limited and a new SSTable is created. A merging compaction that rewrites all SSTables into a single SSTable is called as major compaction. These major compactions allow bigTable to reclaim all the resources used by deleted data. The deleted data is removed from the system at regular intervals, which is crucial when we store sensitive data.

**Refinements:** Compression, caching for read performance, locality groups, bloom filters, commit log implementation, speeding up tablet recover are some of the refinements possible with Bigtable.

**Related Works:** C-store and bigtable share many characteristics in common; shared nothing architecture

.