UNI: kb2896 October 3, 2016
Name: KUNAL BAWEJA
COMS E6998 Cloud Computing and Bigdata (Fall-2016)

# PAPER REVIEW: BIGTABLE - A DISTRIBUTED STORAGE SYSTEM FOR STRUCTURED DATA

## Overview

Bigtable is a reliable and highly scalable distributed storage system for managing structured data. It can handle petabytes of data across thousands of commodity servers. It provides clients with dynamic control over layout and format, in turn providing high performance, low latency solution for a variety of applications. Bigtable has successfully achieved scalability, high performance, applicability across varied situations and high availability.

Bigtable resembles databases closely, as it's objective is to optimize storage and performance with respect to structured data. It doesn't support a full relational data model and rather its provides clients with a simple data model which supports dynamic control over data layout and format. Data indexing is done using rows and columns and clients can carefully set database parameters in schemas to choose whether to serve their data from memory or disk.

## Data Model

Bigtable is a sparse, distributed and persistent multi-dimensional sorted map, having key value pairs. In this map, keys are comprised of a row key, a column key and a timestamp and this composite key maps to an uninterpreted array of bytes(uninterpreted string).

- **Rows**: The row keys are arbitrary strings, typically 10-100 bytes, but going up to 64KB in size. Every data read and write operation under a single row key is atomic irrespective of the number of different columns being written or read within the same row, which makes it easier for clients to make concurrent updates to different columns of same row. Bigtable maintains data sorted lexicographically on row key. Each row range is called a *tablet* which is the unit of distribution and load balancing in bigtable.
- **Column Families:**These are the sets of column keys, which forms the basic unit of access control and disk memory accounting in Bigtable. It is used to compress data and must be created before any data can be stored under any column key of the given family.
- **Timestamp:** Each cell in a Bigtable can have multiple versions of the same data and these different versions are indexed by timestamps, which are represented as 64 bit integers in Bigtable.

## API Overview

The Bigtable API provides functions to create and delete tables and column families and change cluster, table and column family metadata(access control). Client applications can read or delete values from individual rows or iterate over any subset of data(reads) within Bigtable. It supports an array of features:

- **Single-Row Transactions**: Perform *atomic read-modify-write sequences* on data stored *under a single row key*.
- **Integer Counters:** Individual storage cells can be used as integer counters

- **Client Scripts**: Supports execution of client supplied Bigtable scripts within server address space.

## Construction

Bigtable has been built on top of *Google File System (GFS)*, which is used to store log and data files in this case. Similar to *Google File System*, a Bigtable cluster operates over a cluster of storage devices and performs its computations over a shared pool of machines, which are running a variety of applications.
Bigtable relies on a cluster management system and a distributed lock service, called *Chubby*, for scheduling jobs, managing resources on shared machines, dealing with machine failures and monitoring machine status.

## Implementation

Bigtable Implementation comprises of three major components viz. a client linked library, a master server and multiple tablet servers. As the names suggest, each client interacts with Bigtable via the Bigtable library calls, all calls for request of data operations are made to the master server which then directs the appropriate tablet servers to service those requests. Similar to *Google File System's* chunkservers, the tablet servers of Bigtable can join or leave a system at any given point of time.

The master node assigns tablets to tablet servers, detects addition or removal of tablet servers, performs load balancing and garbage collection of files in underlying *Google File System*.

Similar to *GFS* the client data is communicated directly between tablet servers and requesting clients, however in this case most clients never interact with the master node of Bigtable as the clients do not need to ask the master node for tablet location information.

## References

- Chang, Fay and Dean, Jeffrey and Ghemawat, Sanjay and Hsieh, Wilson C. and Wallach, Deborah A. and Burrows, Mike and Chandra, Tushar and Fikes, Andrew and Gruber, Robert E., "Bigtable: A Distributed Storage System for Structured Data", in Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, OSDI '06, (Seattle, WA, USA) pp.15-15, ACM, 2006
- S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03, (New York, NY, USA), pp. 29–43, ACM, 2003.