[3]. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS). 2008 Jun 1;26(2):1-26.

The problem being addressed?

With the widespread adoption of the Internet, users are utilizing the features of various products developed by Internet companies such as Google. This has led to the demand for a resilient, fault-tolerant storage system to handle incoming structured data from applications. The authors Ghemawat et al. studied the problem of structuring and versioning huge volumes of data that accumulates in Google data centers from various applications.

Limitations of Existing work?

The authors mentioned that projects like Boxwood aim to provide a distributed infrastructure for building higher-level services, such as databases and file systems, but that the protocols for handling structured data were not present. This made the existing system not highly beneficial. Similarly, several vendors created parallel databases such as the Oracle Real Application Cluster Database, which uses shared disks and distributed lock managers to handle data. In the same way, IBM DB2 Parallel Edition also uses a shared-nothing architecture similar to Bigtable, but they handle data in the context of relational databases. All the discussed databases and file systems do not directly support client applications that wish to store huge data or they must be tailored as per Google's growing data needs.

Summary of Proposed Strategy?

The authors proposed BigTable, a distributed structured storage system that meets the capacity and scaling needs of Google. It is a sparse, distributed, persistent, and multi-dimensional sorted map that is indexed using a timestamp, column key, and row key as a unique string of an uninterrupted array of bytes. Initially at Google,

it was used to perform web page indexing, and referred to as a web table. Each row has a size of 64 KB, and all the unique strings identified in a table are lexicographically sorted and stored in a tablet. A Tablet is data structure that holds multiple rows as a group for better row locality of reference. Similarly, columns are grouped into sets called column families where access control and memory accounting are performed, and are denoted by the following syntax (family: quantifier). Timestamps are used for indexing to create multiple versions of the same data.

BigTable uses the Google File System (GFS), a distributed file system that stores data across multiple clusters, in the form of log and data files. Typically Big Table relies on cluster management system software for scheduling jobs and monitoring machine status. Internally, every row is stored in the Google SSTable file format. A persistent ordered immutable map of keys and values with each block of size 64 KB (which is configurable) that loads Block Index into memory to search for other blocks. A Block Index is a special index in Bigtable which contains metadata that maps data blocks to their locations in a distributed file system.

BigTable relies on Chubby, which is a highly available distributed lock service that has five active replicas and uses the Paxos algorithm to maintain the consistency of its replicas. One server is elected as master among the pool and other nodes are annotated as tablet servers that have tablets.

BigTable maintains a three-level hierarchy similar to a B+ tree as a) Root tablet, b) Metadata tablets, and c) User tables to store tablet location information.
The first level is a file stored in Chubby that contains the location of the root tablet.

The root tablet maintains the location of all tablets in a special metadata table. Each metadata tablet contains the location of user tablets, and the root tablet itself is the first tablet in a metadata table. Similarly, all the tablets that are tracked are present in the user table (where all the data rows that are inserted are present) with reference to the metadata table.

Data records are persistently stored in GFS using the SSTable file format, which stores redo records in a commit log. Recently committed ones are stored in the primary memory of a tablet server and is referred to as a memtable (An in-memory sorted buffer).

When a client initiates a read request, the tablet servers that are provisioned by the master node acquires an exclusive lock on a uniquely named file in a specific chubby directory. If there are any network connectivity issues, it retries the connection until it reobtains the lock, and later on, it will terminate itself if the file path does not exist in Chubby.

Similarly, when the tablet server does not respond to master messages, it tries to acquire a lock on the chubby file and if it succeeds then it is live. Otherwise it implies the tablet server is dead. Therefore it will be replaced by the master and reassigns all of its tablets back to unassigned state. Similarly, all the write operations also follow the sequence of steps. Big table is currently running in production and used by many products internally in Google.

Summary of Methodology and Result of Evaluation?

The authors conducted several tests on Bigtable using a cluster of N tablet servers. Each tablet server was configured to use 1 GB of memory and a GFS cell consisting of 1786 cells, with two 400 GB IDE hard drives each. The tests were

conducted to evaluate the performance of Bigtable and its ability to handle a large volume of data. The results showed that Bigtable was able to handle petabytes of data and provided high availability, scalability, and fault tolerance.

Any Research Directions Provided?

The authors proposed various research directions, including novel distributed file system methodologies that utilize shared-nothing architecture to provide fault tolerance and high availability.