COMPSCI 590S Systems for Data Science **Fall 2016**

Lecture 13: Google File System, BigTable

Lecturer: Emery Berger *Scribe(s): Arjun Sreedharan, Yuchen Cai*

13.1 Google File System

13.1.1 Lease

There need to be a mechanism to control client access to chunks. The naive approach here would be to use locks. But, locks are not tolerant to failures. Since there is a possibility that the client can die before releasing a lock it acquired, the Master will need to store all modifications made by the client and rollback them in the event of a failure. Instead GFS uses leases. Lease is a time limited granting of exclusive control of a chunk to a client. Once this lease expires, the Master is free to hand over the lease to another client. Therefore right before committing to a chunk, the client has to check again if it still holds the lease. Leases are also used in Bigtables.

13.1.2 Record append

Record append is an atomic append operation provided by GFS. The client specifies only the data and it is appended to the end of the file. All writes go to exactly one of the many replicas. It establishes an order of replicas for the chunks to be written to, and therefore maintains deterministic ordering. Record append assumes idempotent operations. Therefore, the same update can happen more than once. It follows a weakened append semantics - not everything is in order, not compulsorily one after another, but makes sure data is consistent and not scrambled.

13.2 Bigtable

Bigtable is a sparse, persistent, distributed NoSQL data storage developed by Google.
Apache equivalent of Bigtable: HBase.

13.2.1 Data model

Generally, NoSQL datadases store data as simple *key* $\rightarrow$ *value* pairs. Bigtable's data model is a generalization of such data storage systems. Here, the map is indexed by a row key, column key, and a timestamp: *(row:string, column:string, time:int64)* $\rightarrow$ *(value:string)*

Column1	Column2	...	ColumnN	Value

Fig: A logical overview of Bigtable rows

A general key-value store may emulate this by jamming all columns to create a composite key. Bigtables do not do this because the user may only be interested in the data in only some of the columns, and therefore a more performant lookup is possible. Bigtables are sparse data stores, which means they are expected to have a lot of cells with no data. Using timestamps, different versions of a cell are maintained in most recent first order. The number of timestamped versions depend on a threshold set. The threshold specifies either that only the last n versions of a cell be kept, or that only new enough versions be kept. The columns are grouped as families, and each column is identified as a pair *(family: qualifier)*. The row IDs are sorted and split into different groups called *tablets*. This lets BigTable take advantage of locality of data. This model however performs poorly on updates. It follows a *write few read many model*.

13.2.2 Tablet

Each tablet contains one or more SSTables. SSTable (Sorted String Table) is a file format that provides an immutable, ordered key-value map. An SSTable contains multiple blocks (or chunks) and an index to locate the blocks. These chunks are of size 64KB each. The SSTable index stores ranges of rows and is loaded into the main memory for fast lookup. SSTables use binary search within the block for finding rows. Additionally, the recently committed updates are stored in memory in a sorted buffer called *memtable*. When the memtable size reaches a threshold, it is frozen and converted to an SSTable. Bigtable uses a distributed lock service named Chubby for handling synchronization issues. Chubby runs on the Paxos algorithm and uses the lease mechanism.

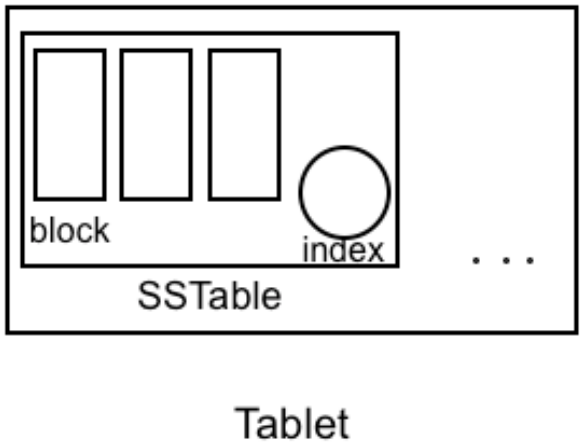

Figure 13.1: Tablet structure

13.2.3 Block map

When there are a lot of tablets, looking up tablets is a difficult task. To accomplish this, Big table uses a three-level block map. This works very similar to how a file system maps a file to its blocks.