Eric Johnson
efj2106

This paper was a nice follow-up to the GFS paper that we read last week because it was the logical next iteration in Google's development of a scalable database system – instead of focusing on the data management and replication BigTable functions on top of and in tandem with GFS to deliver a distributed storage system for managing **structured data** designed to scale. This paper comes in 2006, when several of Google's core serves such as Google Earth, Google Finance, and Google Analytics. The ultimate bread-and-butter of BigTable is its ability to manage latency (backend bulk processing to real-time data serving). Interestingly enough the system was designed to manage petabytes worth of data (a spectrum we are just now beginning to enter with Big Data solutions) but an order of magnitude Google was already familiar with in the mid-2000s.

Much like a giant set of SQL relational database tables BigTable is more clever in the sense that it does not supper a full relational data model, instead it provide clients with a simple data model that provides dynamic control over data layout and format. The client can also specify if the data is served out of memory or disk. The fundamentals of BigTable are a distributed multidimensional sorted map – a map indexed by a *row key*, *column key*, and *timestamp*.

(row:string, column:string, time:in64) -> string

A simple example of how this functions was given for Webtable – a caching technique use for websites. A web url would be the row key, various aspects of the website would be column names, and the contents would be columns associated under timestamps when it was fetched. The component that really allows the system to achieve a distributed nature is how it manages and stores row ranges. A "row range" is called a tablet which is the unit of distribution and load balancing. Again going back to webtable this means that sub sites of the same web domain are stored together in the same **tablet** (all of the pages under columbia.edu) would appear in the same tablet, making retrieval and analytics on that domain easier because the data is stored nearby. Column keys are grouped into **column families** which is just another organizational level to columns. This makes logical sense when talking about thinks like language because all of the columns associated with a particular language of a domain can be stored and grouped together locally instead of mixing multiple columns of different languages. The garbage-collection mechanism was designed to store only the three most recent versions of every page. Within this whole system Google SSTable (a way to manage immutable map from keys to values is used.

I think an interesting aspect to the paper I didn't know about before was that MapReduce (the process used in a Hadoop to perform large jobs/analytics across distributed data and merge the intermediate results as needed) was actually developed at Google. (I had no idea)

On top of all of this they implement a rather clever distributed lock service called **Chubby** which uses one master and several replicas to manage atomicity of the writes/reads to BigTable. If you think about it in a hierarchical b-tree fashion the Chubby service is responsible for assigning tablets (row ranges) to different tablet servers, which themselves will manage balancing of tablet-server load and garbage collection in GFS. (Chubby – Master Node > Root Table >

MetaData Tablets > UserTables) The Root Table contains and manages the locations of all of the tablets in the different tablet servers. The master is responsible for detecting when a tablet server is no longer serving its tablets. Similar to how a master node in GFS manages the chunk servers and the replica of each piece of data this master node manages the tablet servers underneath it. The paper goes on to discuss how the data is compressed and moved between tablet servers as needed to manage load and replication.

In the end I would say this is very interested paper which also helped to shape how big data structured data is stored. IBM and several commercial providers offer similar solutions called BigSQL which help manage the distribution and replication of structured data. Interesting takeaways from the paper were the authors comments on proper system-level monitoring and how had this been done earlier they would have been able to determine the source of their bottlenecks faster. I think this was a valuable observation we all should use when developing and implementing distributed systems.