Anastasia Shpurik

Database HW #22

*Document#1 (BigTable: Confusing)*

**Bigtable**

-a distributed storage system for managing structured data that is designed to scale to a very large size

-used by Google Earth, Google Finance, Orkut, Google Analytics

-wide applicability, scalability, high performance, high availability

-supports dynamic control over data layout & format, allows clients to reason about locality properties of data represented in underlying storage

-data is indexed using row and col names (arbitrary strings)

**Data Model**

-Bigtable is sparse, distributed, persistent multidimensional sorted map

(row:string, column:string, time:int64) → string

-each read/write of data under single row key is atomic

-tablet: row range (unit of distribution & load balancing)

-column keys grouped together into sets called 'column families' – basic unit of access control

-each cell in Bigtable can contain multiple versions of same data – these are indexed by timestamp

-Bigtable relies on a highly-available & persistent distributed lock service called Chubby

-consists of 5 active replicas, one of which is elected to be the master and actively serve requests

-Bigtable uses Chubby to ensure that there is at most one active master at any time,to store the boostrap location of Bigtable data, to discover tablet servers and finalize tablet server deaths, to store Bigtable schema info, and to store access control lists

**Implementation**

-3 components: library that is linked to every client, 1 master server, many tablet servers

-each tablet is assigned to one tablet server at a time

*Document #2 (BigTable: Better)*

-at its core, BigTable is a map

-the data you put in map persists after the program that created or accessed it is finished

-built upon distributed filesystems so the underlying file storage can be spread out among an array of independent machines

-BigTable key/value pairs kept in string alphabetical order

-EX: table whose keys are domain names: domains all near each other

-essentially a map of a map

-each key points to map with certain number of keys. The mappings of these inner keys are “Column Families”

-table's col families specified when table is created – difficult to modify later

-a column family can have any number of columns

-all data is versioned either using an integer timestamp or another integer of your choice

-sparseness – row-based gaps (gaps between keys), multiple columns in each column family, or none

*Document #3 (Apache HBase)*

-Hbase is open source, distributed, sorted map datastore modeled after Google's BigTable

-not relational database (very light “schema”)

-consists of rows, each has primary key, each row may have any num of columns, rows sorted

-different sets of columns may have different properties & access patterns

-some API: get(row), put(row, Map<column, value>), scan(key range, filter), increment, etc..

-region: subset of table's rows, like range partition

-regionServer: slave, serves data for reads & writes

-master: responsible for coordinating slaves, assigns regions, detects failures, controls admin functions

|  |  |  |
| --- | --- | --- |
|  | RDBMS | HBase |
| Data layout | row-oriented | column-family-oriented |
| Transactions | Multi-row ACID | Single row only |
| Query Lang | SQL | Get/put/scan/etc.. |
| Read/Write throughput lim | 1000s queries/sec | Millions queries/sec |

-Hbase is ordered range partitions, not hash (comparing to key-value)

-also, sparse column storage (not key-value)

-use Hbase if: you need random write, random read, or both

-need to do many thousands of operations per sec

-don't use Hbase if: you append to your dataset and tend to read whole thing

-data is not huge