Bigtable: A Distributed Storage System for Structured Data

Presented by: Chaitali Mulay (cmulay)

Motivation

System requirements:

- Handle varying data size
- Deal with different workloads
- Scalability
- Reliability and Availability
- Performance
- Recovery from Failures

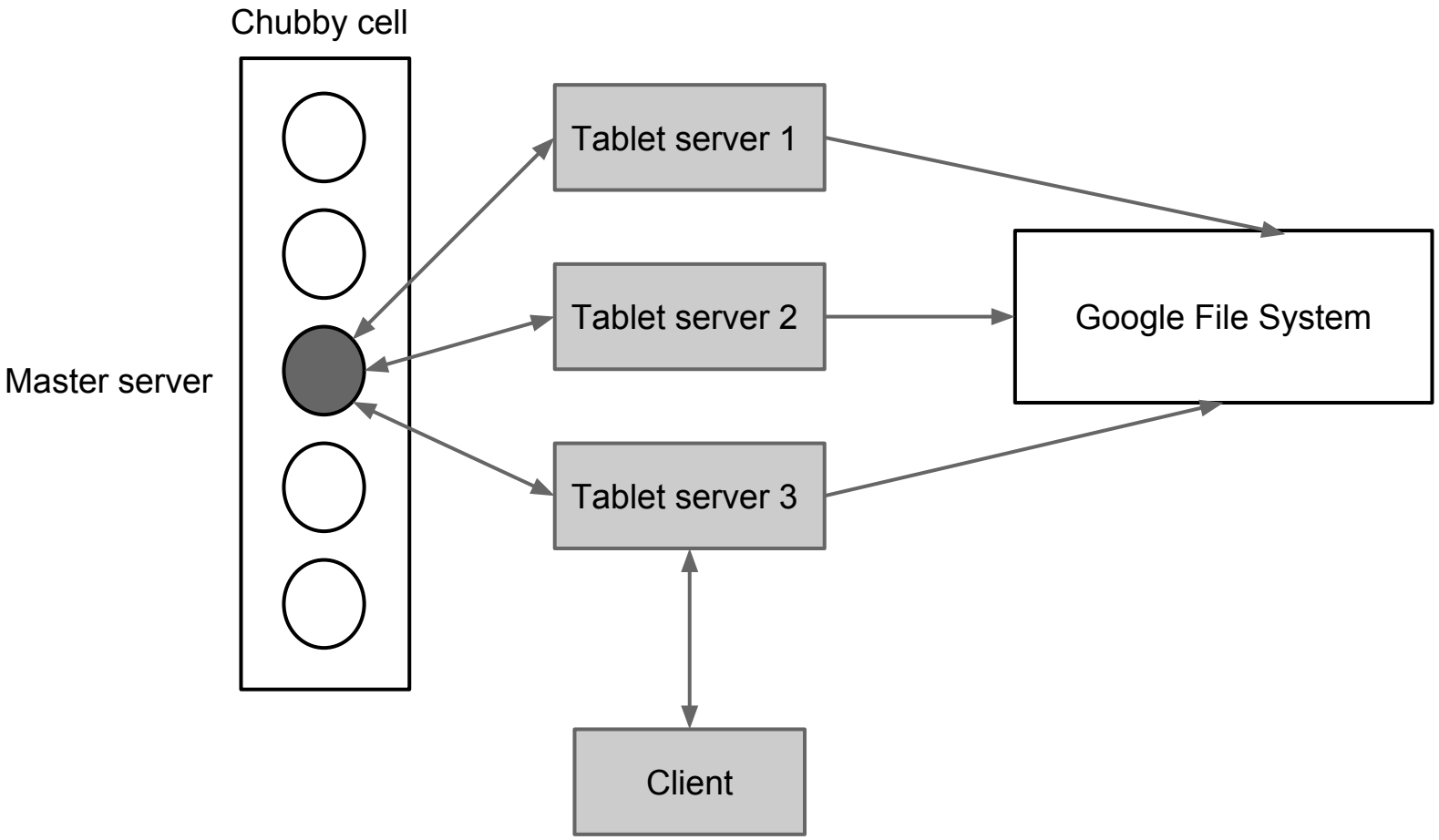
Data Model

- Not a full relational model
- Key - Value pairs stored in *SSTables*
- Data indexed using row key, column key and timestamp
- Row range is dynamically partitioned into ***tablets***
- Benefits
  - Simple
  - Control over layout and format
  - Take advantage of locality properties

Important components

- Master server
- Tablet servers
- Library linked to clients
- Google File System (store data and logs)
- Chubby distributed lock service (elect master and other services)
- Cluster management system
- Memtable (sorted memory buffer for recent updates)

High-Level Visual Overview



Scalability

- Google File System can run on commodity hardware and is highly scalable
- Tablet servers can run on similar hardware and can be incrementally added

Reliability and Availability

- Chubby distributed lock service
  - to elect master server
  - to store Bigtable schema information
- Consistency of replicas is maintained using Paxos Algorithm
- Issues
  - If Chubby service fails, Bigtable fails.
  - Solution?
- Several tablet servers
- Cluster management system

Performance

- Data is stored in *SSTables* (lexicographically sorted)
- Reads within small ranges are quick
- Block index in *SSTables* allows lookup with single disk seek
- Clients communicate directly with *tablet servers*
- Clients prefetch and store tablet locations in cache

Recovery from Failures

- Chubby service ensures there is one live master server
- Chubby keeps track of live and expired tablet servers
- Tablet server loses lock if it expires
- Tablets assigned to expired tablet servers are moved to unassigned groups
- Memtable and SSTables store updates that can be used for recovery

In Summary

- System with single master architecture that is reliable and highly available
- Reliability and availability comes from use of Chubby
- Good performance due to simple data model and clients directly communicating with tablet servers
- Google File System is highly scalable and fault tolerant

Derivatives and Inspirations

- Spanner (by Google) is layered on Bigtable
- LevelDB (by Google) is inspired by Bigtable
- Apache's Cassandra uses Bigtable's data model
- Apache HBase is Bigtable-like system layered on HDFS (Hadoop Distributed File System)

Some questions

- Why not use a full relational schema?
- Should Bigtable have multi-row transactions?
- What improvements are possible?
  - Hardware-level
  - Software-level