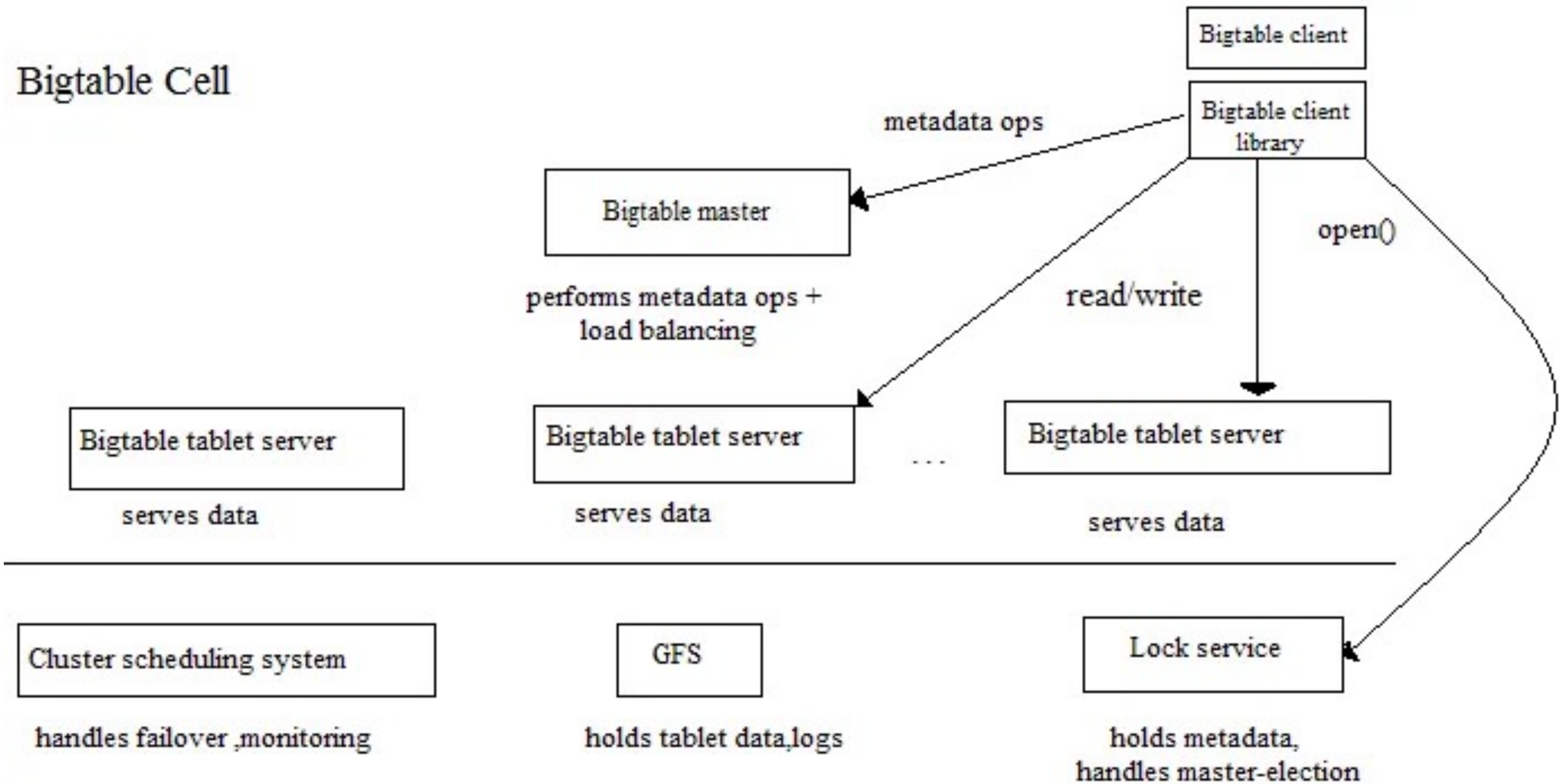
Bigtable

大数据分析 | 何铁科

http: // hetieke.cn

南京大學
NANJING UNIVERSITY

" BigMap"


Bigtable Cell
Bigtable client
Bigtable client library
metadata ops
Bigtable master
performs metadata ops + load balancing
read/write
open()
Bigtable tablet server
serves data
Bigtable tablet server
serves data
. . .
Bigtable tablet server
serves data
Cluster scheduling system
handles failover ,monitoring
GFS
holds tablet data,logs
Lock service
holds metadata, handles master-election

References

- Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. 2008. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.* 26, 2 (Jun. 2008), 1-26

- Bigtable: A Distributed Storage System for Structured Data, Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI), November 2006

- Lecture slides from Cornell University – Advanced Distributed Storage Systems course

Roadmap

1. Motivation
2. Overview
3. Data Model
4. Overview of Client API
5. Building Blocks
6. Fundamentals of Bigtable implementation
7. Refinements
8. Conclusions

Motivation(I)

- **Lots of (semi-)structured data at Google**
  - URLs:
    - Contents, crawl metadata, links, anchors, pagerank, …
  - Per-user data:
    - User preference settings, recent queries/search results, …
  - Geographic locations:
    - Physical entities (shops, restaurants, etc.), roads, satellite image data, user annotations, …
- **Scale is large**
  - billions of URLs, many versions/page (~20K/version)
  - Hundreds of millions of users, thousands of q/sec
  - 100TB+ of satellite image data

Motivation(II)

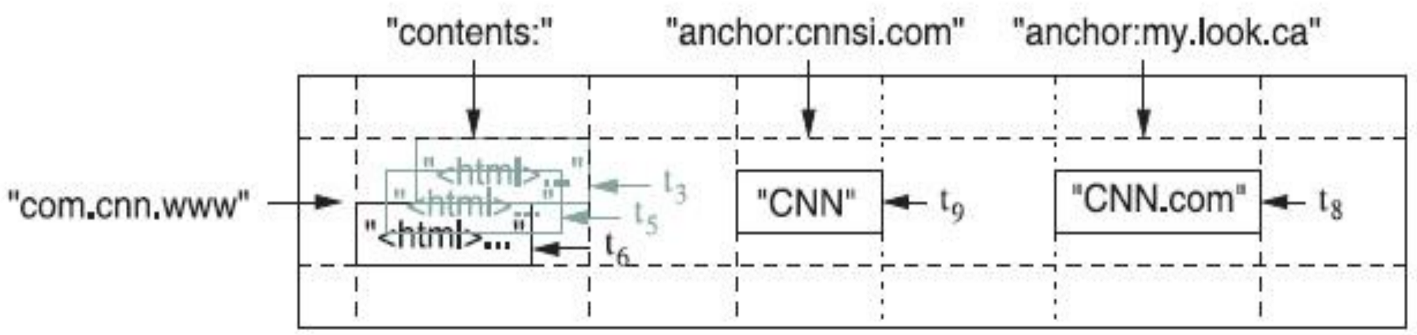
- Required DB with wide scalability, wide applicability, high performance and high availability
- Cost of commercial data bases
- Building system internally would help in using it for other projects with low incremental cost
- Low level storage optimizations can be done, which can be helpful in boosting performance

Overview

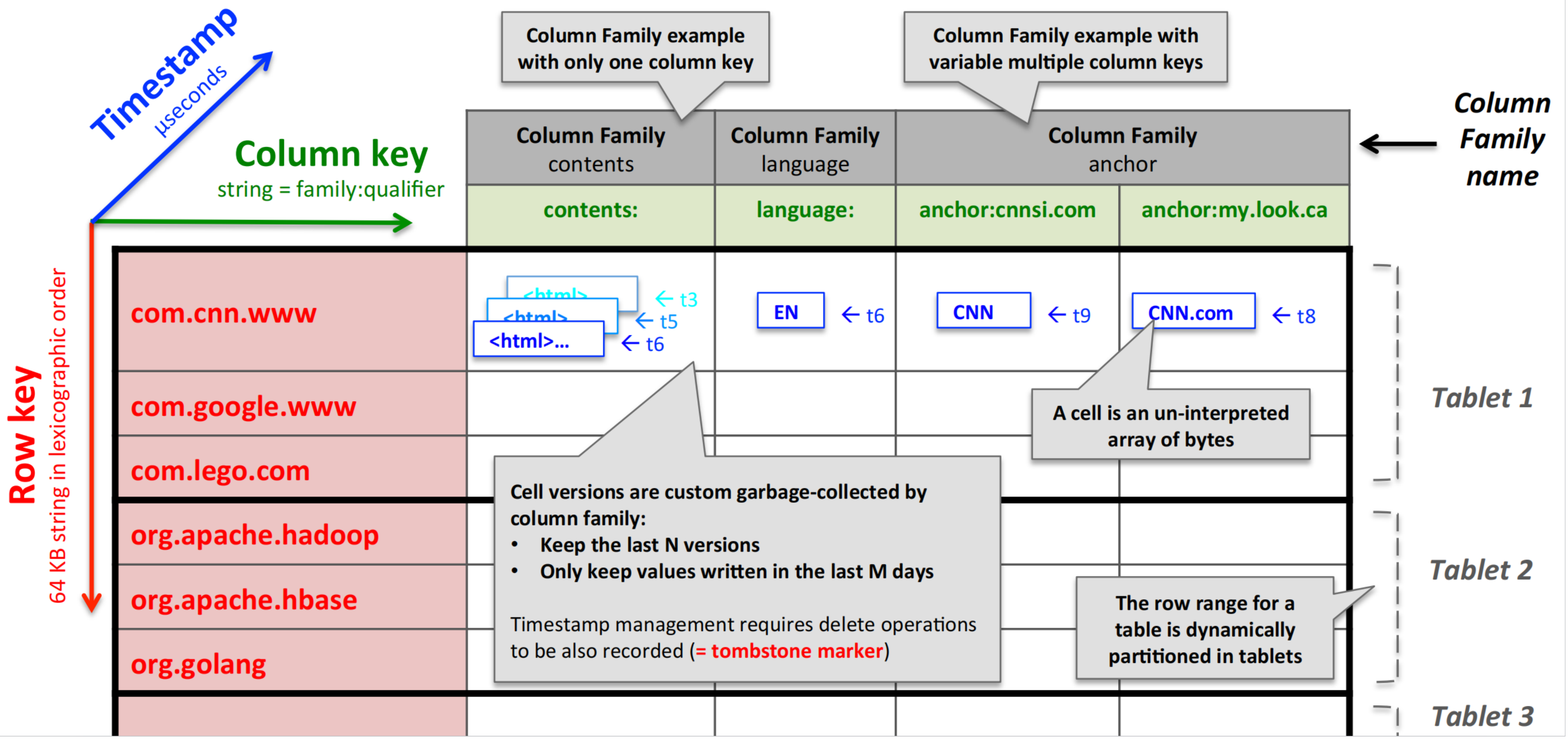
- Bigtable does not support full relational data model
- Supports dynamic control over data layout and format
- Clients can control locality of their data through choice of schema
- Schema parameters let client dynamically control whether to serve data from memory / disk.

Data Model (I)

- Distributed multi-dimensional sparse map
- (Row, Column, Timestamp) -> Cell contents
- Row keys are arbitrary strings
- Row is the unit of transactional consistency



Data Model (II)



Data Model (III)

- **Rows with consecutive keys are grouped together as “tablets”.**
- **Column keys are grouped into sets called “column families”, which form the unit of access control.**
- **Data stored under a column family is usually of the same type.**
- **Column key is named using the following syntax:**
  family :qualifier

Data Model (IV)

- **Access control and disk/memory accounting are performed at column family level.**
- **Each cell in Bigtable can contain multiple versions of data, each indexed by timestamp.**
- **Timestamps are 64-bit integers.**
- **Data is stored in decreasing timestamp order, so that most recent data is easily accessed.**

Client APIs(I)

- **Bigtable APIs provide functions for:**

- Creating/deleting tables, column families
- Changing cluster , table and column family metadata such as access control rights

Client APIs(II)

- **Bigtable APIs provide functions for:**

- Support for single row transactions
- Allows cells to be used as integer counters
- Client supplied scripts can be executed in the address space of servers

Building Blocks (I)– underlying Google infrastructure

- **"Chubby" for the following tasks**
  - **Store the root tablet, schema information, access control lists.**
  - **Synchronize and detect tablet servers**

- **What is Chubby ?**
  - **Highly available persistent lock service.**
  - **Simple file system with directories and small files**
  - **Reads and writes to files are atomic.**
  - **When session ends, clients loose all locks**

Building Blocks (II)

- GFS to store log and data files.
- SSTable is used internally to store data files.
- What is SSTable ?
  - Ordered
  - Immutable
  - Mappings from keys to values, both arbitrary byte arrays
  - Optimized for storage in GFS and can be optionally mapped into memory.

Building Blocks (III)

- **Bigtable depends on Google cluster management system for the following:**
  - **Scheduling jobs**
  - **Managing resources on shared machines**
  - **Monitoring machine status**
  - **Dealing with machine failures**

Implementation(I) - Master

- **Three major components**
  - **Library (every client)**
  - **One master server**
  - **Many tablet servers**

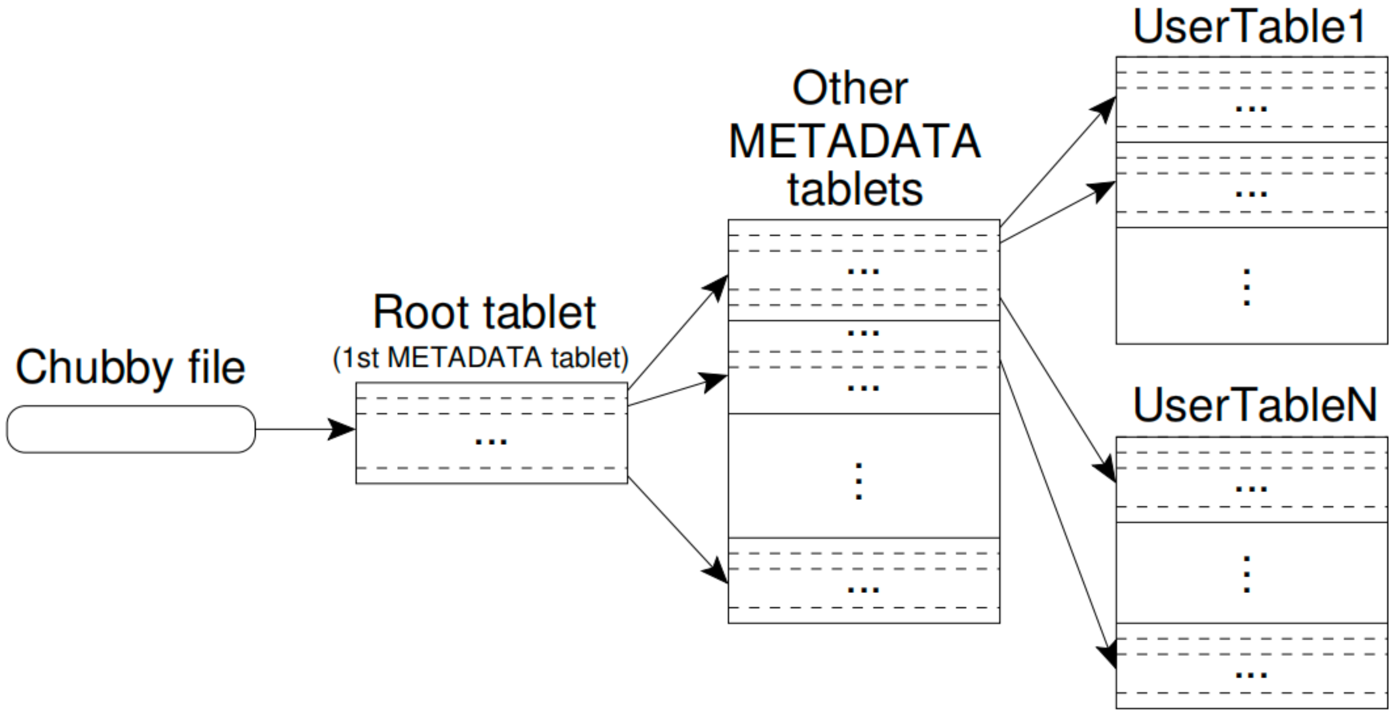
- **Single master tasks:**
  - **Assigning tablets to servers**
  - **Detection the addition/expiration of servers**
  - **Balancing servers' loads**
  - **Garbage collection in GFS**
  - **Handling schema changes**

Implementation – Tablet Server

- Tablet server tasks:
  - Handling R/W requests to the loaded tablets
  - Splitting tablets
- Clients communicate with servers directly
  - Master lightly loaded
- Each table
  - One tablet at the beginning
  - Splits as grows, each tablet of size 100-200 MB

Tablet Location

- We use a three-level hierarchy analogous to that of a B+-tree to store tablet location information



Tablet Location

- 3-level hierarchy for location storing
  - One file in Chubby for location of *Root Tablet*
  - Root tablet contains location of *Metadata tablets*
  - Metadata table contains location of user tablets
    - Row-Key: [Tablet's Table ID] + [End Row]
- Client library caches tablet locations
  - Moves up the hierarchy if location N/A

Tablet Assignment

- Master keeps track of assignment/live servers
- Chubby used
  - Server creates & locks a unique file in *Server Directory*
  - Stops serving if loses lock
  - Master periodically checks servers
    - If lock is lost, master tries to lock the file, un-assigns the tablet
  - Master failure do not change tablets assignments

Tablet Assignment

- Master restart
  - Grabs unique master lock in chubby
  - Scans server directory for live servers
  - Communicate with every live tablet server
  - Scans Metadata table

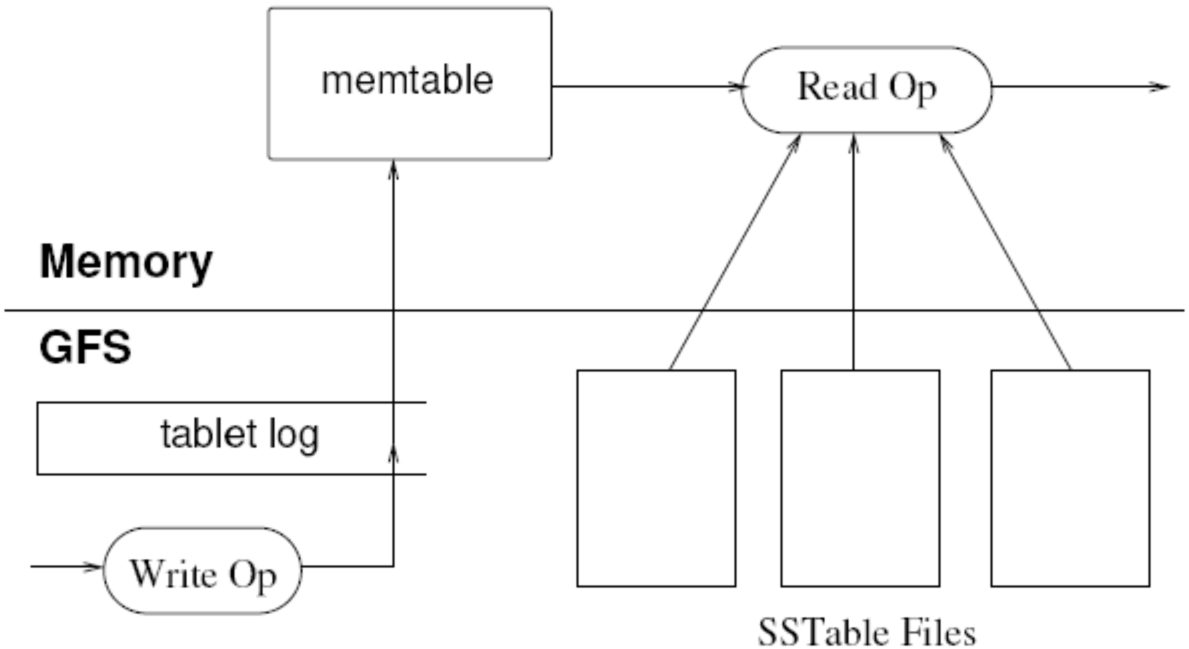
Tablet Changes

- Tablet Created/Deleted/Merged → master
- Tablet Split → tablet server
  - Server commits by recording new tablet's info in Metadata
  - Notifies the master

Tablet Serving

- Tablet Serving
  - Tablets in GFS
  - REDO logs
    - recent ones in memtable buffer
    - Older ones in a sequence of SSTables

Tablet Serving



Tablet Recovery:

- Server reads its list of SSTables from METADATA Table
- List = (Comprising SSTables + Set of ptrs to REDO commit logs
- Server reconstructs the status and memtable by applying REDOs

R/W in Tablet

- Server authorizes the sender
  - Reading list of permitted users in a chubby file
- Write
  - Valid mutation written to commit log (memtable)
    - Group commits used
- Read
  - Executed on merged view of SStables and memtable

Compaction

- Minor compaction
  - (Memtable size > threshold) → New memtable
  - Old one converted to an SSTable, written to GFS
    - Shrink memory usage & Reduce log length in recovery

Compaction

- Merging compaction
  - Reading and shrinking few SSTables and memtable
- Major compaction
  - Rewrites all SSTables into exactly one table
  - BT reclaim resources for deleted data
  - Deleted data disappears (sensitive data)

Refinements – Locality Groups

- Client groups multiple col-families together
- A separate SSTable for each LG in tablet
- Dividing families not accessed together
  - Example
    - (Language & checksum) VS (page content)
  - More efficient reads

Refinements – Locality Groups

- Tuning params for each group
  - An LG declared to be in memory
  - Useful for small pieces accessed frequently
    - Example. Location Column Family in Metadata

Refinements – Compression

- Client can compress SSTable for an LG
- Compress format applied to each SSTable block
  - Small table portion read wout complete decomp.
- Usually two pass compress
  - Long common strings through large window
  - Fast repetition looking in a small window (16 KB)
- Great reduction (10-1)
  - Data layout (pages for a single host together)

Refinements-Caching for read performance

- Tablet servers use two levels of caching
  - The Scan Cache : high level cache for key-value pairs returned by the SSTable interface to the tablet server code
  - The Block Cache : low level cache for SSTables blocks read from GFS

Refinements - Bloom filter

- Problem: read operation has to read from all SSTables that make up the state of a tablet
  - -Lot of disk access 😭
- Solution: use Bloom filers for each SSTable in a particular locality group
  - – Bloom filter uses a small amount of memory and permit to know if a SSTable doesn't contain a specified row/column pair
  - – Most lookups for non existent rows or columns do not need to touch disk 😊

Refinements-Commit-log implementation

- Commit log implementation
  - Each tablet server has a single commit log
  - Complicates recovery
    - Master coordinates sorting log file *<Table, Row, Log Seq)*

Refinement - Immutability

- Speeding up tablet recovery

- Exploiting immutability
  - Various parts of the Bigtable system have been simplified by the fact that all of the SSTablesgenerated are immutable
  - Garbage collection on obsolete SSTables
  - Immutability of SSTables permit to split tablets quickly

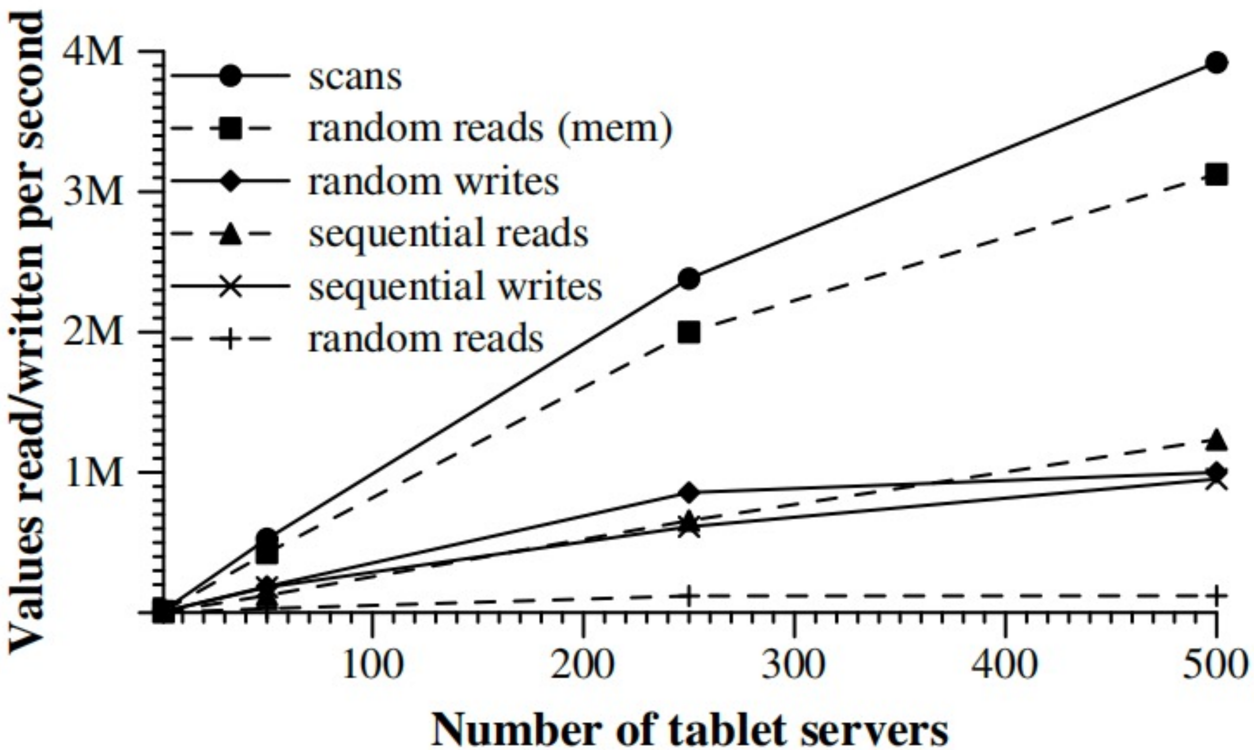
Performance evaluation

- Per-Server #Read/#Write

Experiment	# of Tablet Servers 1	50	250	500
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

Performance evaluation

- Aggregate #Read/#Write



Conclusion

- Bigtable has achieved its goals of high performance, data availability and scalability.
  - It has been successfully deployed in real apps (Personalized Search, Orkut, GoogleMaps, …)

- Significant advantages of building own storage system like flexibility in designing data model, control over implementation and other infrastructure on which Bigtable relies on.