Google Bigtable Database

@ 北京航天航空大学软件学院

18:00 03/20/2012 周二晚 北航主校区 主M201

By 邓侃 Ph.D, 李小吉, 朱小杰
SmartClouder.com

Relational Database Recapitulation

Tweet ID	Time Stamp	Author ID
6793232	2012030618455245	748229
6793231	2012030618455243	481293

Row (Tuple)

Tweet ID	Tweet Content
6793232	我就喜欢这样的挑战文化。
6793231	如果我们的云计算公开课，被砸了场子，那将是我们的荣幸，因为真正的牛人出现了！

Column (Attribute)

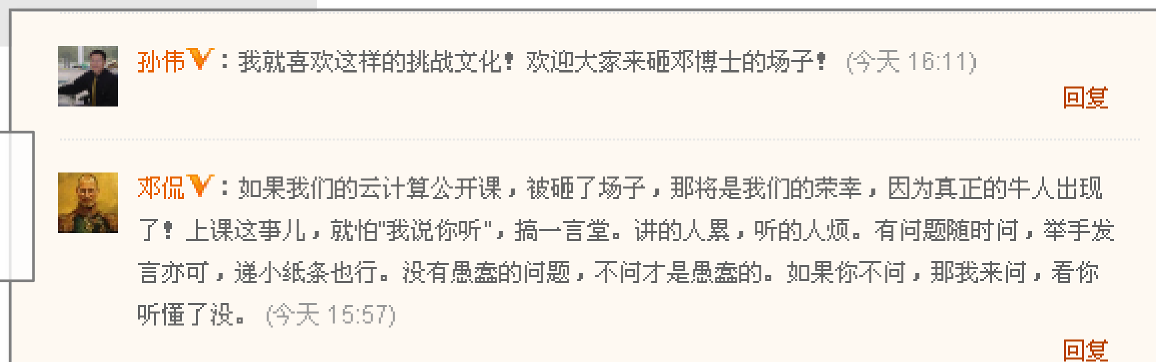
Foreign Key

User ID	Following ID	Follower ID
748229	481293, 223838, …	193922, …
481293	223838, …	748229, 193922, …

Relationship (Table)

Reader ID	Tweet ID in Newsfeed
193922	6793232, 6793231, …
748229	6793231, …

Relational Database for Twitter



Row ID	Index of Columns	Column Len	Column Data	Column Len	Column Data	…
748229	1, 2	8	67932320	24	我就喜欢这样的挑战文化。	…
748230	1, 2	8	67932321	78	如果我们的云计算公开课 …	…

A homebrew general-purposed row physical layer

- SQL DML (Data Manipulation Language):
  SELECT,
  INSERT,
  UPDATE,
  DELETE.

SQL ≈

Relational Database's APIs

Tweet ID	Tweet Content
67932320	我就喜欢这样的挑战文化。
67932321	如果我们的云计算公开课，被砸了场子，那将是我们的荣幸，因为真正的牛人出现了！

- SQL DDL (Data Definition/Description Language):
  CREATE (ALTER, DROP) tablespace,
  CREATE (ALTER, DROP) table,
  CREATE (DROP) view,
  CREATE (DROP) index.

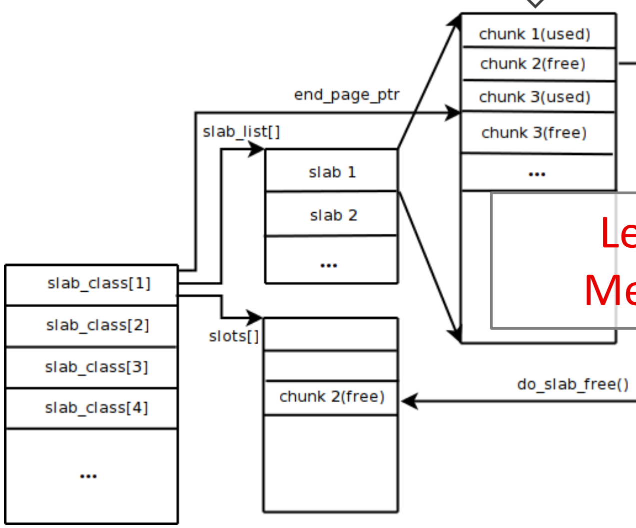
Reader IDa	Tweet ID in Newsfeed
193922	6793232, 6793231, …
748229	6793231, …

Tweet ID	Tweet Content
67932320	我就喜欢这样的挑战文化。
67932321	如果我们的云计算公开课，被砸了场子，那将是我们的荣幸，因为真正的牛人出现了！

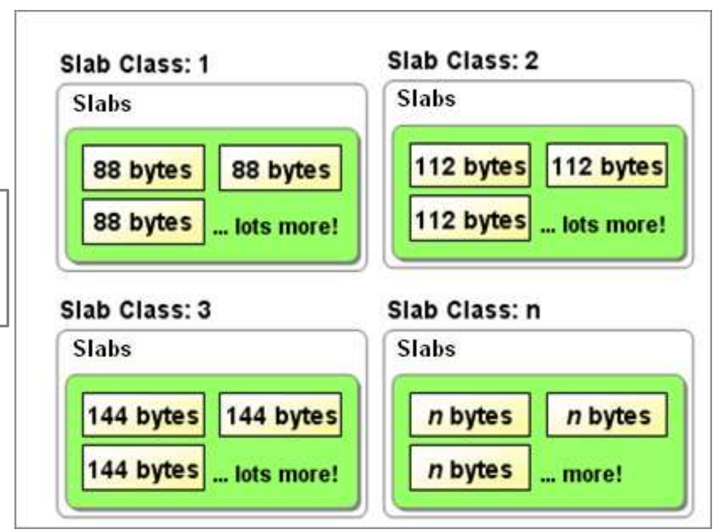
Row data structure

Row ID	Index of Columns	Column Len	Column Data	Column Len	Column Data	...
748229	1, 2	8	67932320	24	我就喜欢这样的挑战文化。	...
748230	1, 2	8	67932321	78	如果我们的云计算公开课 ...	...

Row physical layer



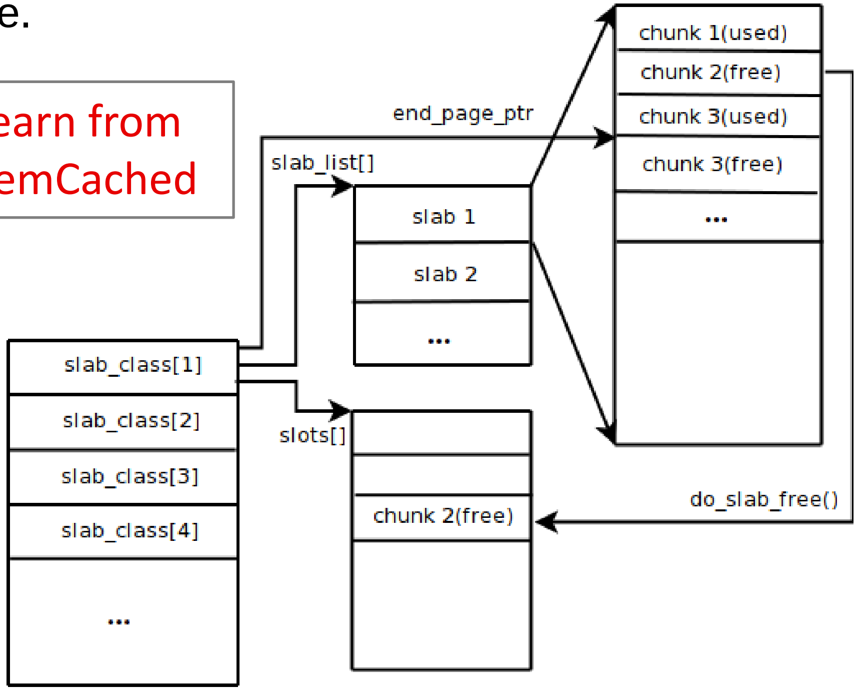
Learn from MemCached



Row ID	Index of Columns	Column Len	Column Data	Column Len	Column Data	...
748229	1, 2	8	67932320	24	我就喜欢这样的挑战文化。	...
748230	1, 2	8	67932321	78	如果我们的云计算公开课 ...	...

- Slab: fixed-size continuous storage space.
- SlabClass:
  A group of slabs of the same size.
  (maybe dispersed)
- Chunk:
  Each slab splits into sequential chunks,
  the chunks are of the same size.
  One chunk consists of many items.
- Slots:
  A list of available chunks.

Learn from MemCached



- Why split the storage into fixed-size slabs and chunks?
  Easy to re-use, but may waste storage space.

Storage Layer

- Writing to disk is slow,

  So, appending is slow, but still much faster than random accessing.

- Store in buffer cache first, then write to disk.

  Store in buffer cache first, then write to disk as log, then merger (commit) into files.

Journaling File System:

1. Speed-up INSERT and UPDATE. 2. Process transaction operations.

Time: T1

Commit Time ⇩

Committed File							Log (Journal)			
Position	#1	#2	#3	#4	#5	#6	@1	@2	@3	@4
Data	10	20	30	40	50	-	Add 2 to #2	Append 90	Del #4	Minus 2 to #2

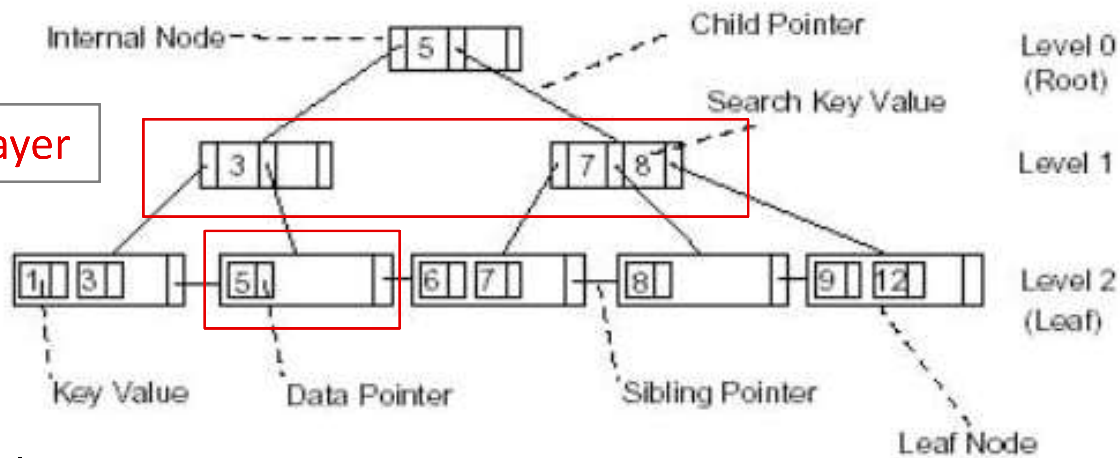
Time: T2

Committed File							Log (Journal)			
Position	#1	#2	#3	#4	#5	#6	@1	@2	@3	@4
Data	10	22	30	-	50	90	Minus 2 to #2	Add 2 to #5	Set #2 80	

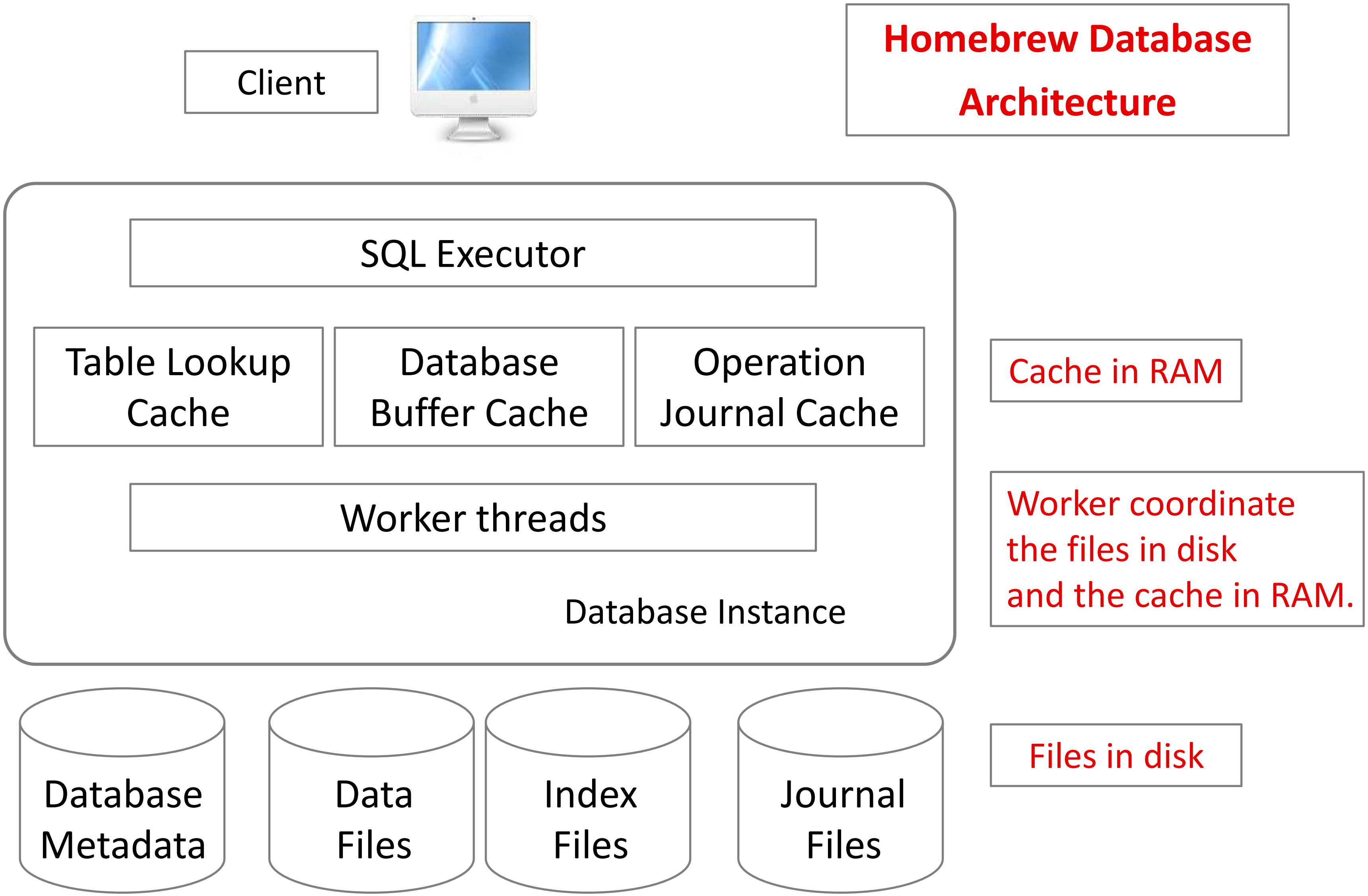
Data(#2) = ? Data(#2) = File(#2) + Log(@1) + Log(@2) + Log(@3) = 80

Number of Keys	Child ID	Key 1	Child ID	Key2	Child ID	…	Sibling ID	Data Pointer
1	(1,3)	3	(5)	-	-	-	-	-
2	(6,7)	7	(8)	8	(9,12)	…	-	-
1	-	5	-	-	-	…	(6,7)	3A17C3

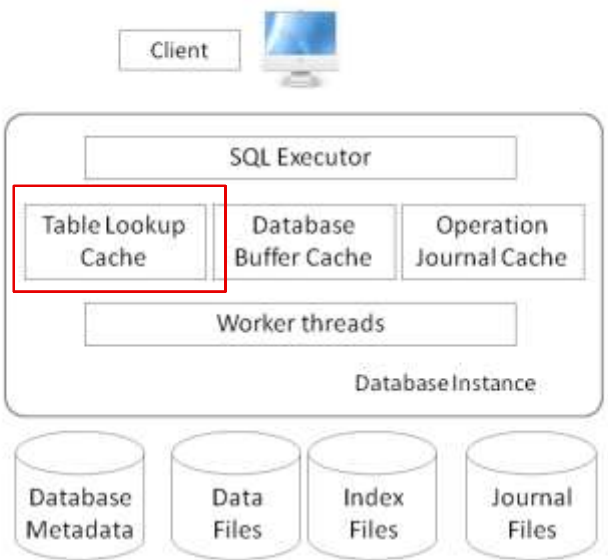
B+ Tree node physical layer



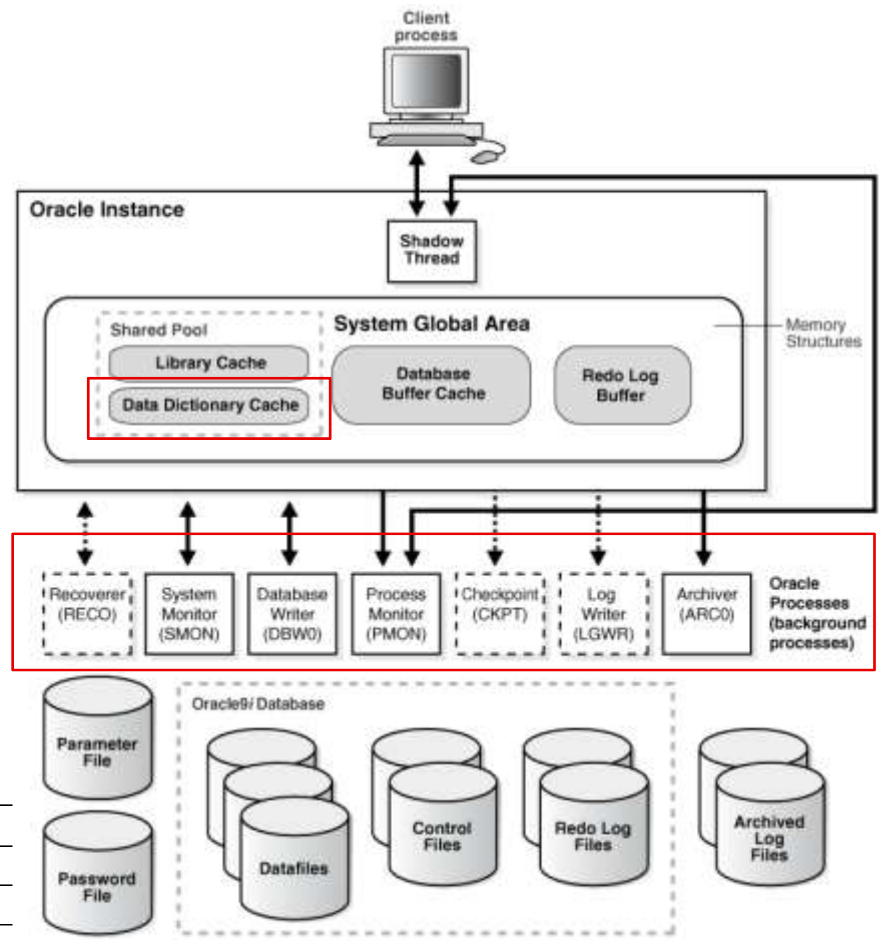
- Two critical database problems:
  1. Transaction, 2. Indexing.
- Transaction can be implemented with Journaling file system.
- Indexing is implemented by B+ tree.
  The physical layer of B+ tree's node is easy to define.


Homebrew Database Architecture
Client
SQL Executor
Table Lookup Cache
Database Buffer Cache
Operation Journal Cache
Worker threads
Database Instance
Cache in RAM
Worker coordinate the files in disk and the cache in RAM.
Database Metadata
Data Files
Index Files
Journal Files
Files in disk

Our Homebrew Database



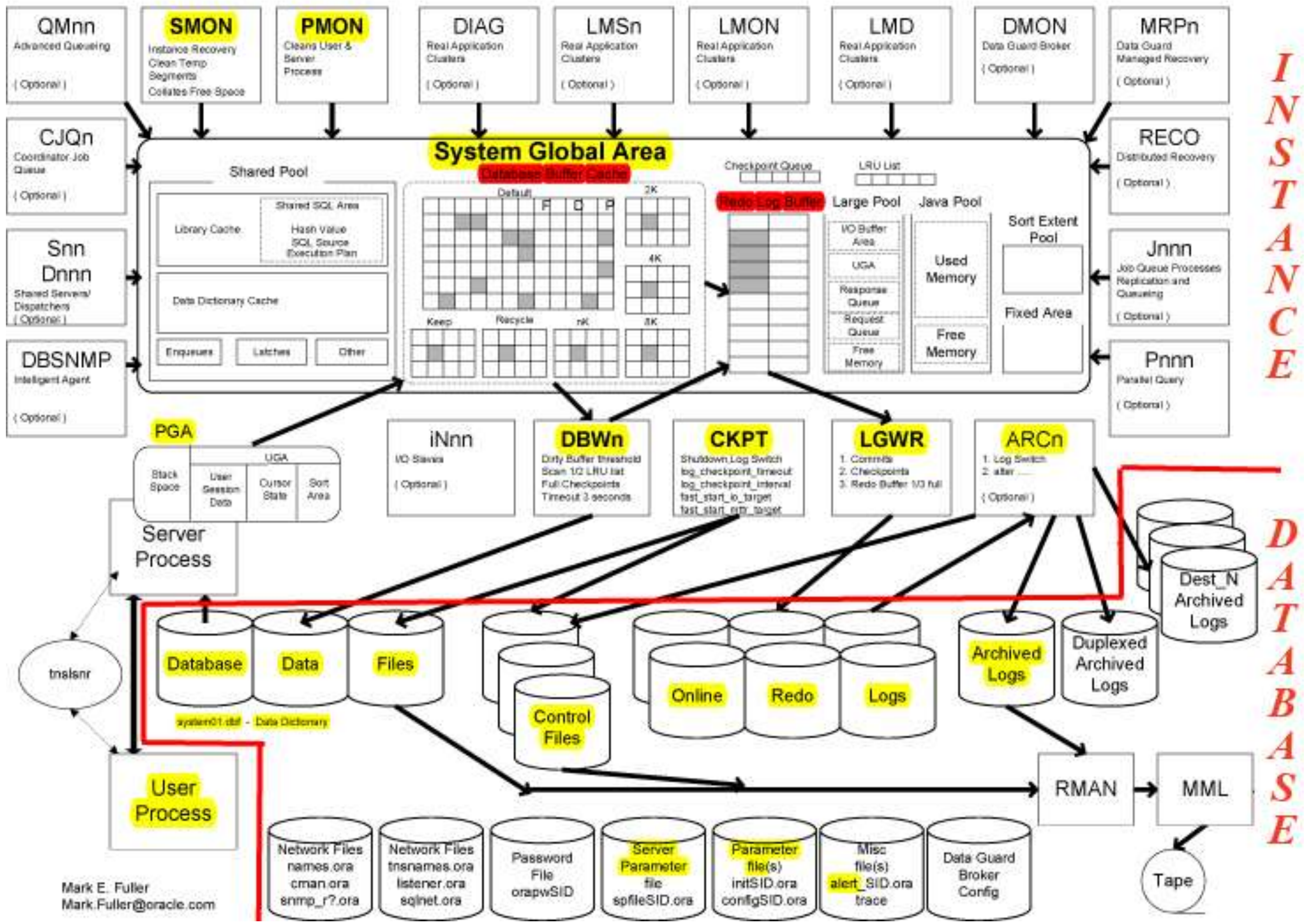
Oracle 9i



Very similar to each other

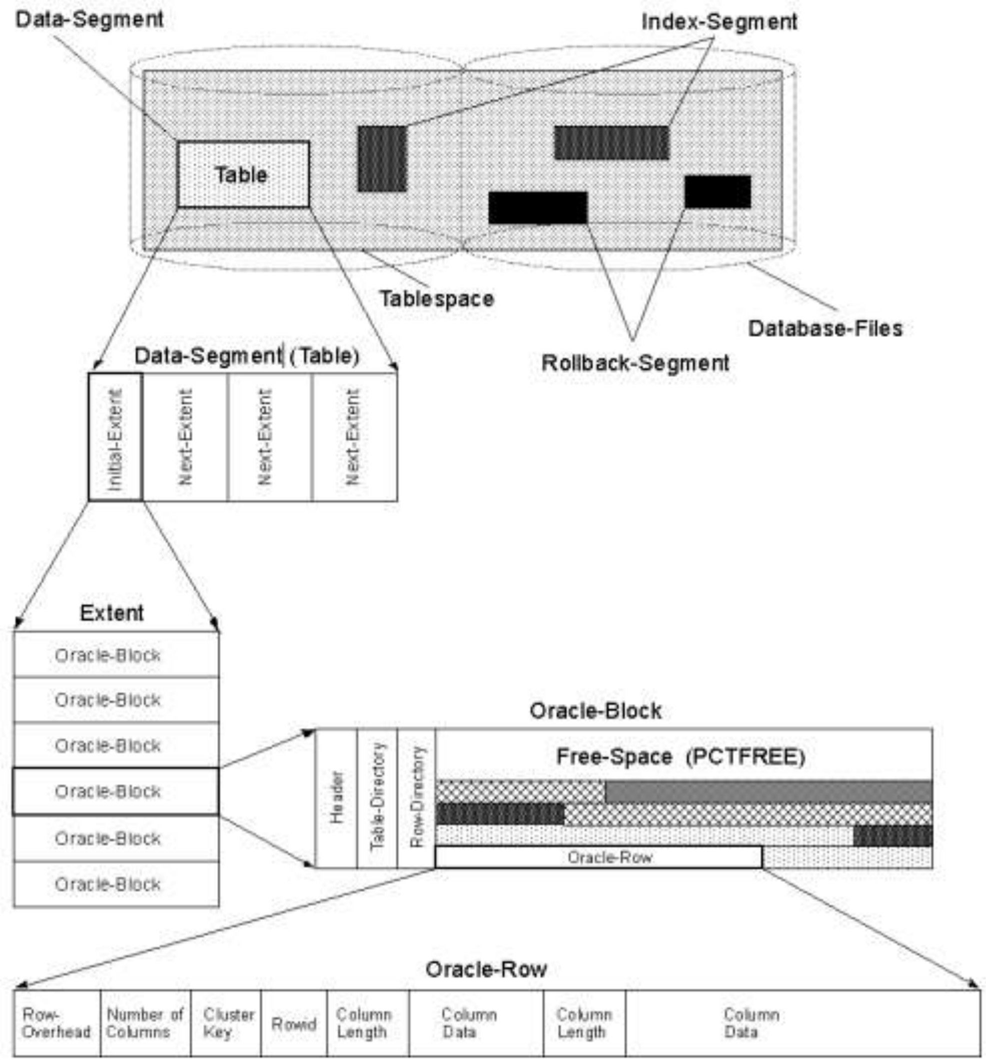
Oracle Thread	Description
DBW0	database writer
LGWR	log writer
PMON	process monitor
SMON	system monitor
CKPT	checkpoint process (or thread on Windows) that runs by default on Windows
ARCH0	archive process (or thread on Windows)
RECO	distributed recovery background process

ORACLE ARCHITECTURE

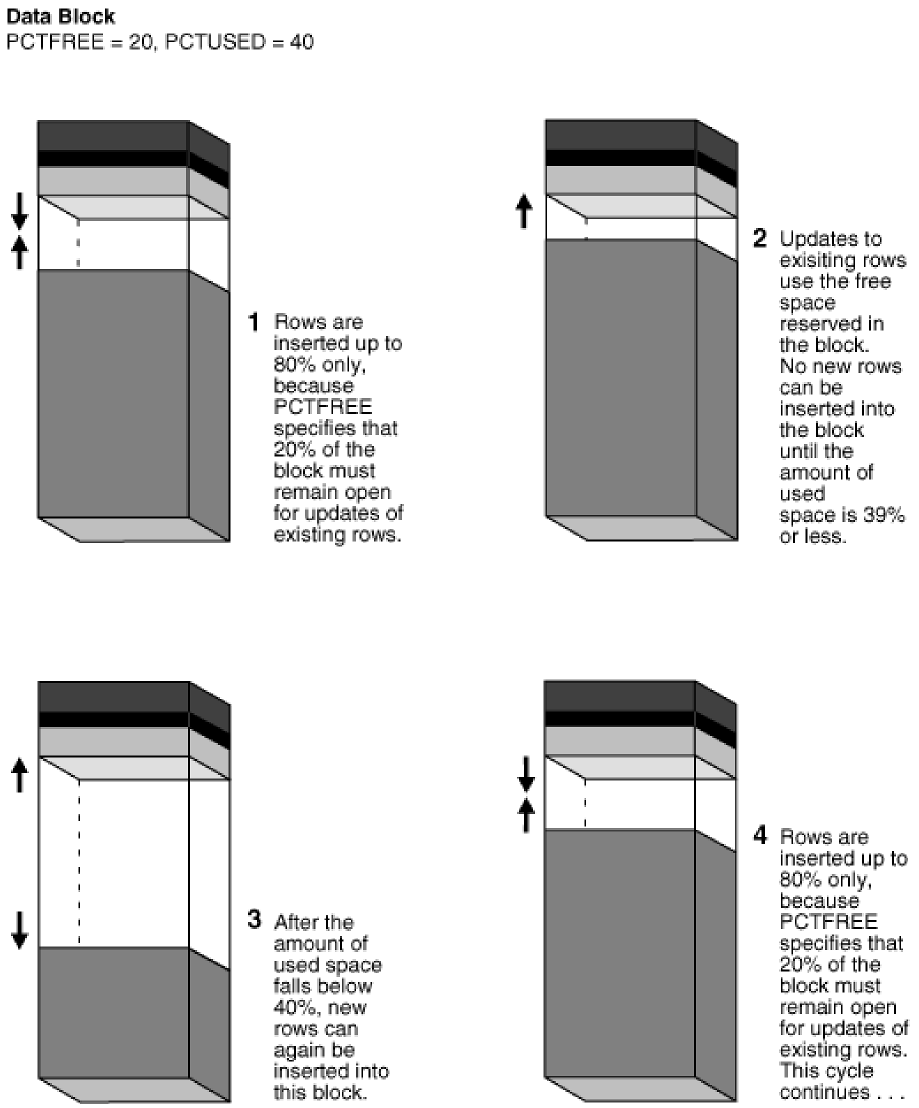


More details, but principally similar to our homebrew.

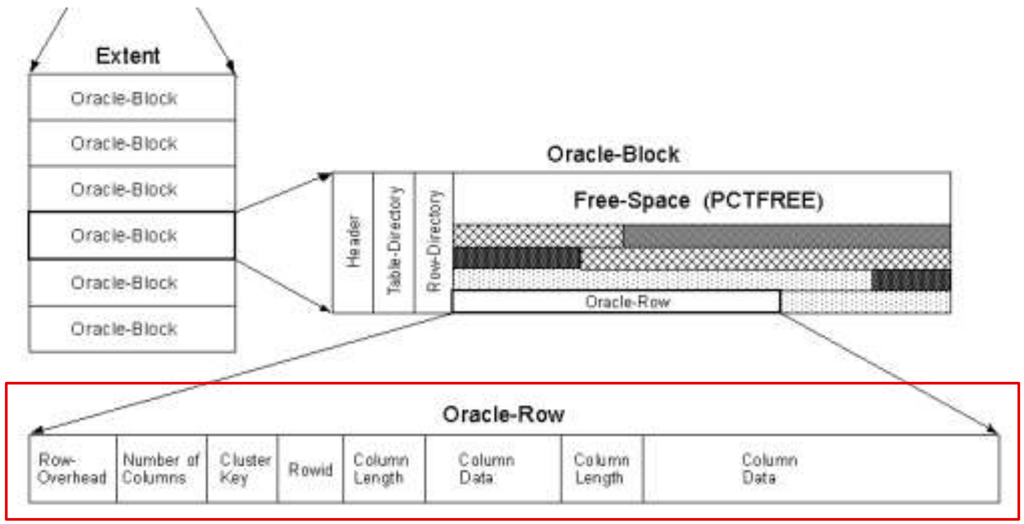
- Blocks:
  a fixed-size storage space,
  contains table rows.
- Extent:
  a continuous disk space,
  contains a group of blocks.
- Table or Index:
  their data stored in
  several extents.
- Disperse:
  The extents of the same table
  may not in the same disk.
  When a table grows with
  more extents are allocated.



- Data block reserves some available space.
- Updating row's values, most likely are in the same block, but may not in the same order.
- Inserting new row, No guarantee in the same block.



- When updating a cell's value, the row's length may be changed.
- Even though the updated row is still stored in the same data block, its position inside the data block may be changed.
- Hence, the rows of a table are NOT stored sequentially in order.
- Inserting makes the disorder and disperse ever further.
- Therefore, data retrieval heavily rely on indexing.



Our homebrew row physical layer, is similar to Oracle's design.

Row ID	Index of Columns	Column Len	Column Data	Column Len	Column Data	…
748229	1, 2	8	67932320	24	我就喜欢这样的挑战文化。	…
748230	1, 2	8	67932321	78	如果我们的云计算公开课 …	…

- Physical layer of a table row:
  A few {column-length, column-value} pairs.

- Physical layer of an B+ indexing tree is similar to that of table row.

- Data block: contains table rows.
  Extent: Physically continuous storage space, containing a group of blocks.
  Segment: a group of extents, may or may not in the same server.
  A Table or Index: consists for multiple segments.

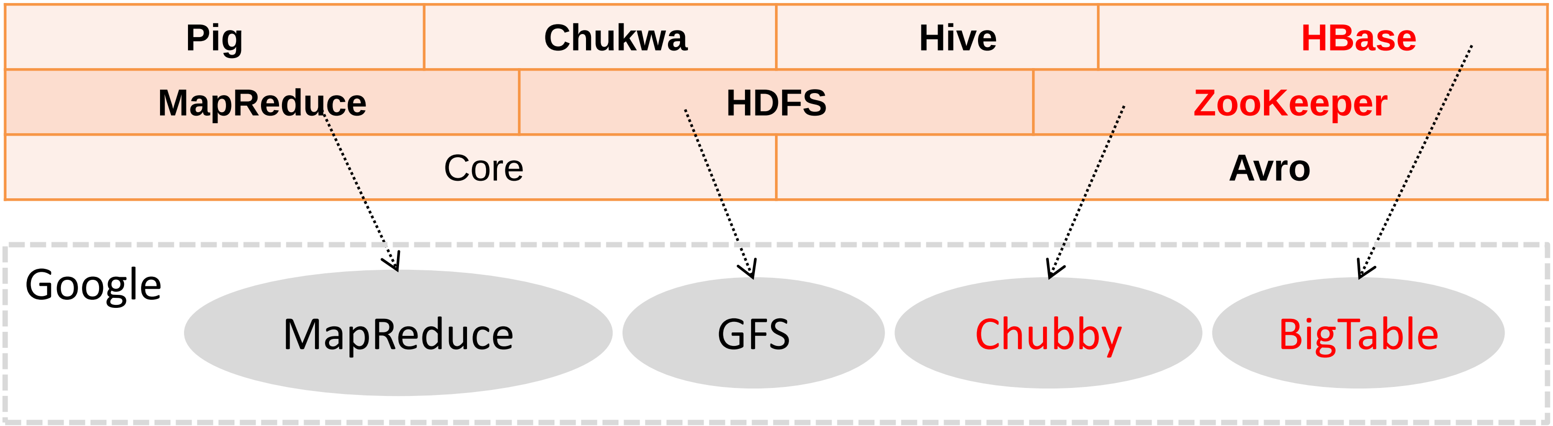
- Insert and update,
  make the data dispersed in the storage space,
  the efficiency of data read/write/update, relies on indexing.

- Cache data in RAM, to speed up data read/write/update.
  Client request → Cache in RAM → Append to journaling log → Merge into data file.

- Transaction is implemented with journaling log.

Hadoop HBase
Google Bigtable

- Hadoop is an open source project, supervised by Apache org. Implemented in Java.
- Hadoop is a distributed system, for large scale storage and paralleled computing. A mimic of Google system.



Hadoop Common: The common utilities that support the other Hadoop subprojects.
Avro: A data serialization system that provides dynamic integration with scripting languages.
Chukwa: A data collection system for managing large distributed systems.
HBase: A scalable, distributed database that supports structured data storage for large tables.
HDFS: A distributed file system that provides high throughput access to application data.
Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying.
MapReduce: A software framework for distributed processing of large data sets on compute clusters.
Pig: A high-level data-flow language and execution framework for parallel computation.
ZooKeeper: A high-performance coordination service for distributed applications.

Row Key	Time Stamp	Column "contents:"	Column "anchor:"		Column "mime:"
"com.cnn.www"	t9		"anchor:cnnsi.com"	"CNN"	
	t8		"anchor:my.look.ca"	"CNN.com"	
	t6	"<html>..."			"text/html"
	t5	"<html>..."			
	t3	"<html>..."			

Logical Structure

- How to handle big tables, with numerous rows and columns? Cut (Shard) into small parts.
- Bigtable is very different from conventional RDBMS's.
- Oracle stores data on ROWs. Bigtable stores data on COLUMNs, more accurately, "Column Families".

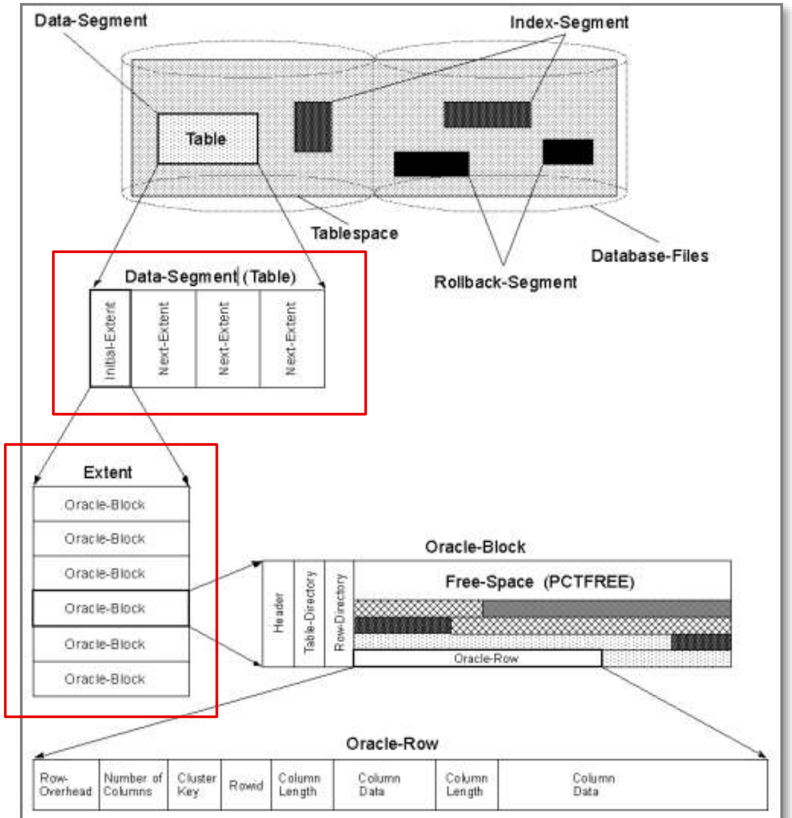
Row Key	Time Stamp	Column "contents:"
"com.cnn.www"	t6	"<html>..."
	t5	"<html>..."
	t3	"<html>..."

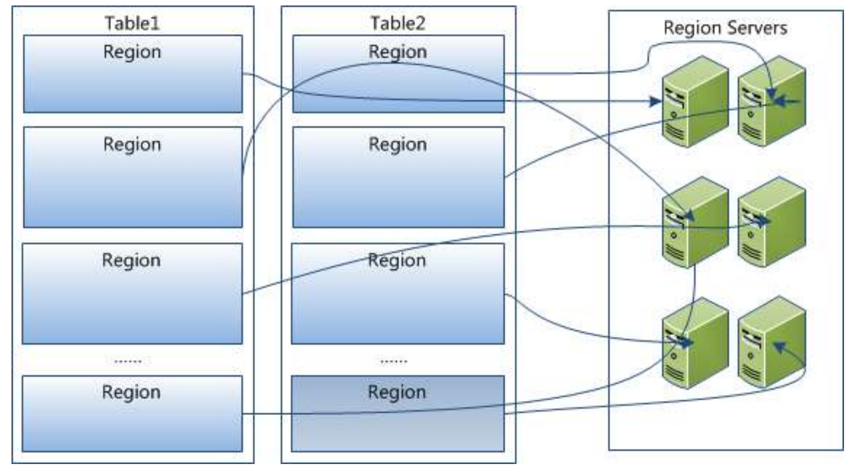
Row Key	Time Stamp	Column "anchor:"
"com.cnn.www"	t9	"com.cnn.www" "CNN"
	t8	"anchor:my.look.ca" "CNN.com"

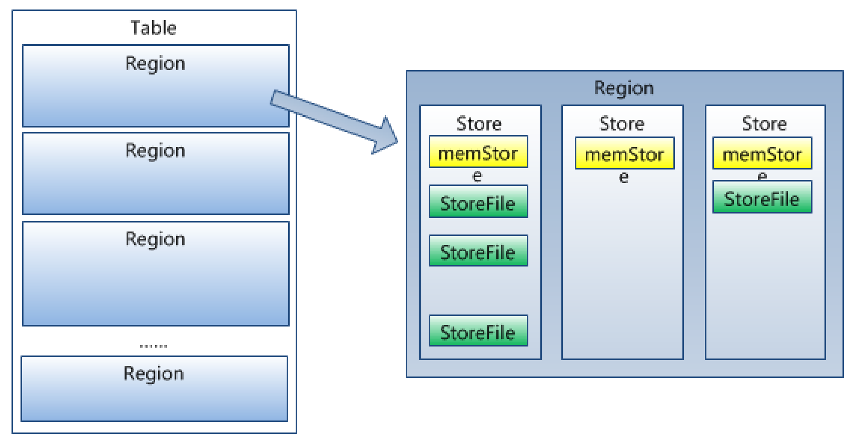
Row Key	Time Stamp	Column "mime:"
"com.cnn.www"	t6	"text/html"

Physical Structure

HBase 0.9 Data Storage







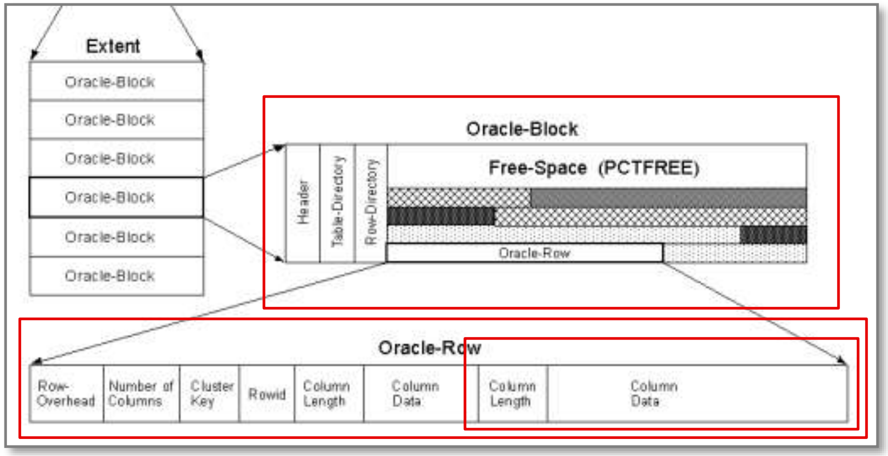
Segment == Table

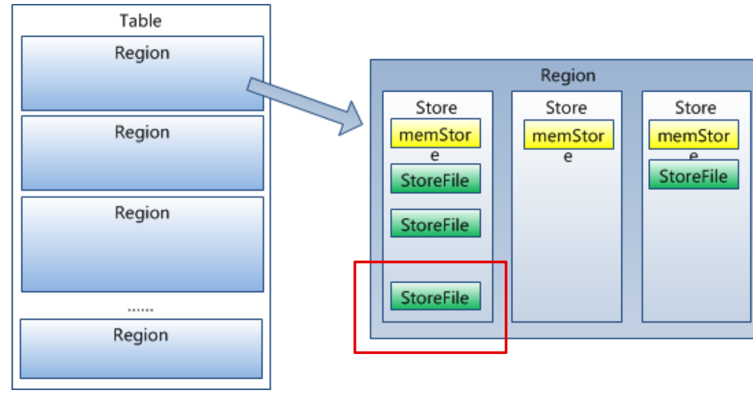
Extent == Region/Store

DataBlock == StoreFile

StoreFile is stored in HDFS as HFile.

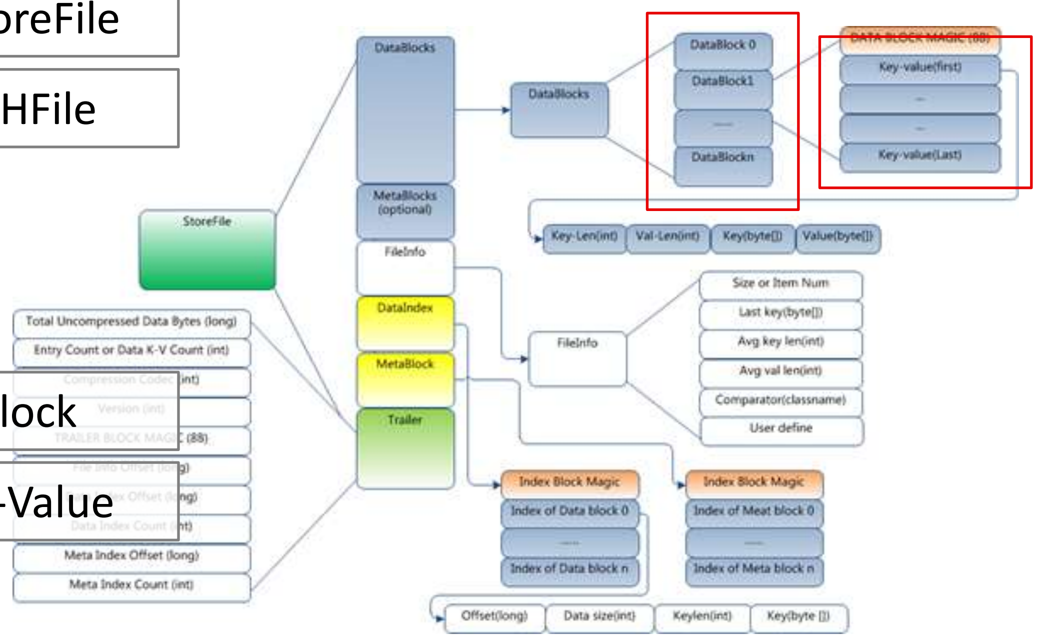
HBase 0.9 Data Storage





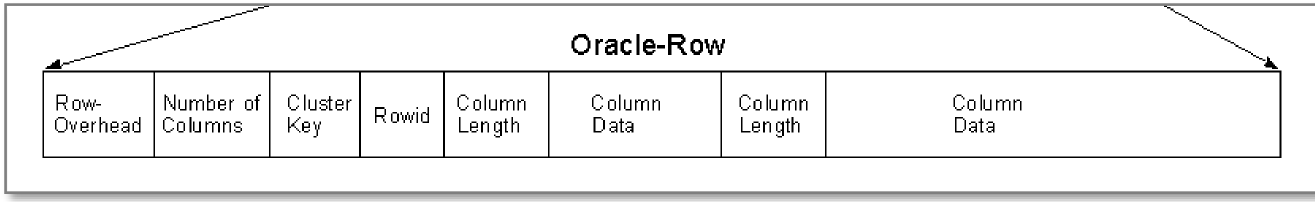
Oracle DataBlock == HBase StoreFile

StoreFile is stored in HDFS as HFile

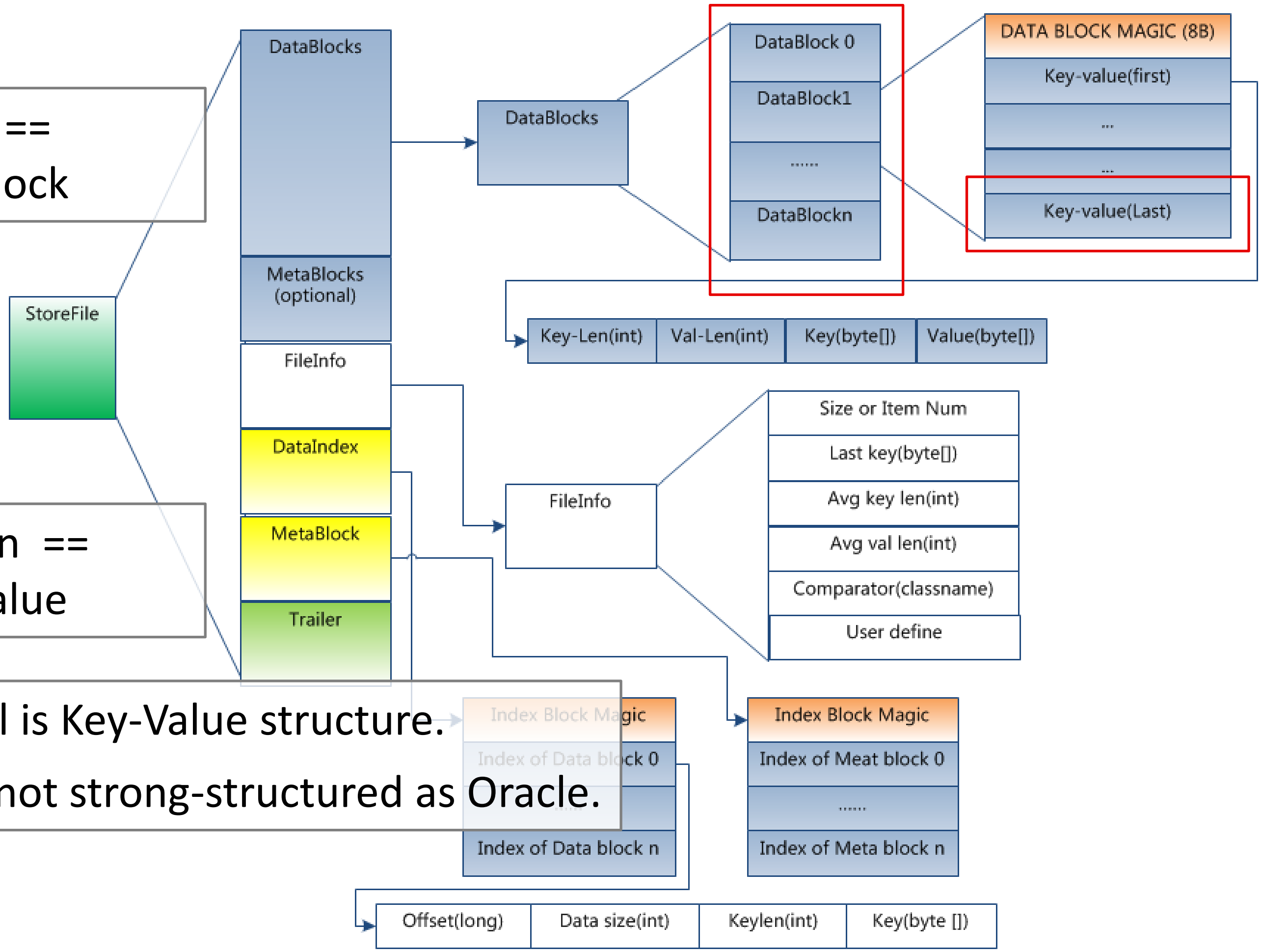


Oracle Row == HBase Datablock

Oracle Column == HBase Key-Value



Oracle Row ==
HBase Datablock

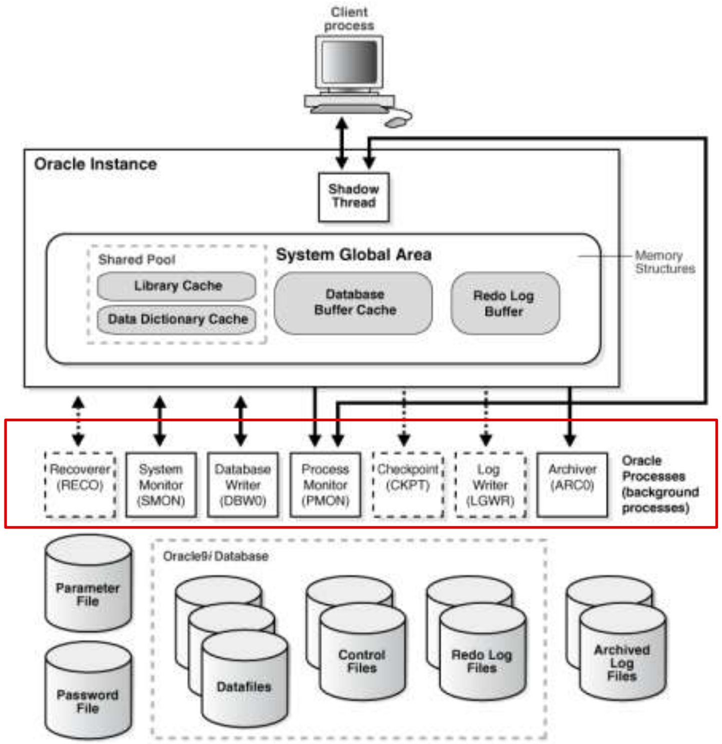


Oracle Column ==
HBase Key-Value

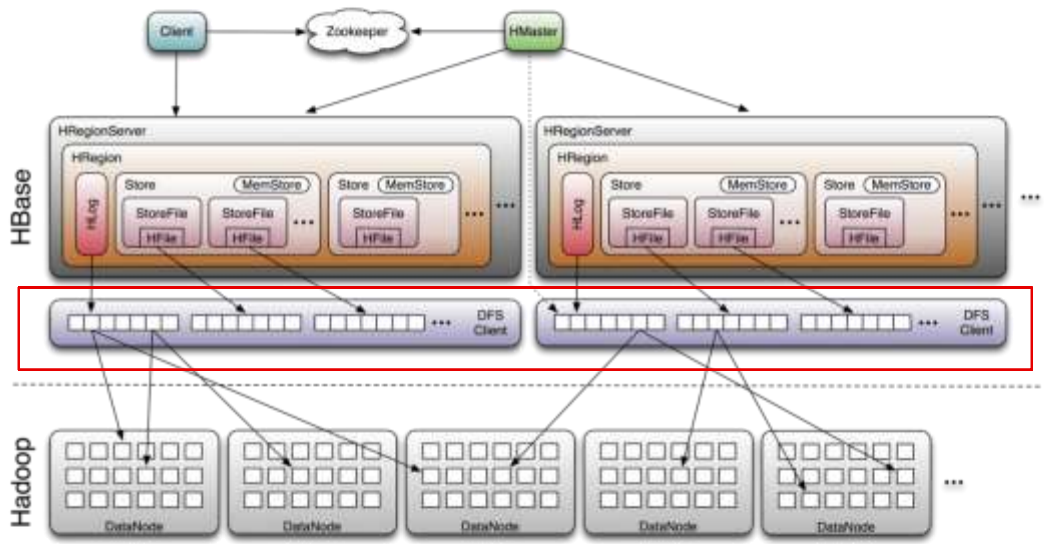
HBase’s cell is Key-Value structure.

So HBase’s table is not strong-structured as Oracle.

Oracle 9i

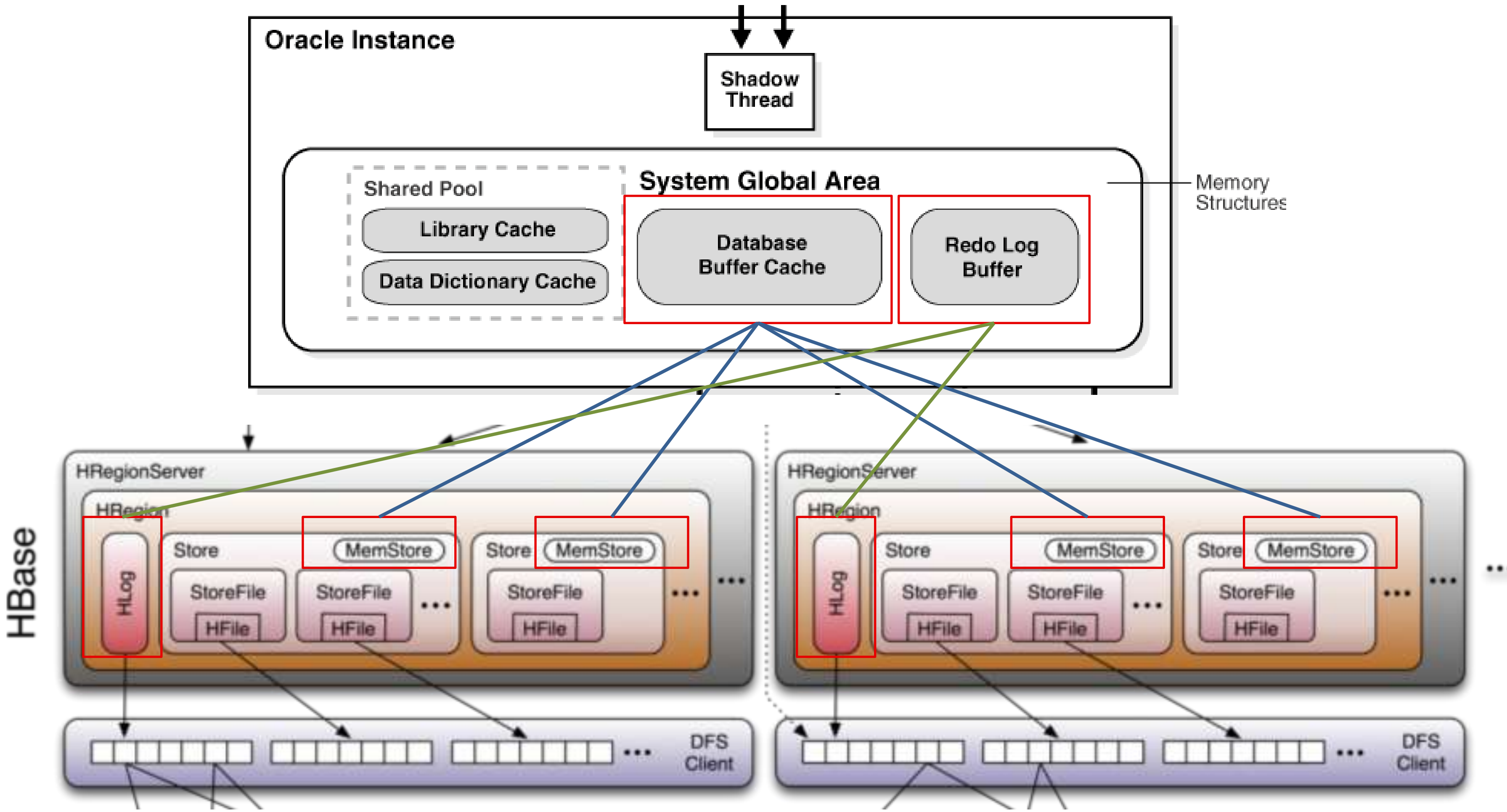


HBase 0.9

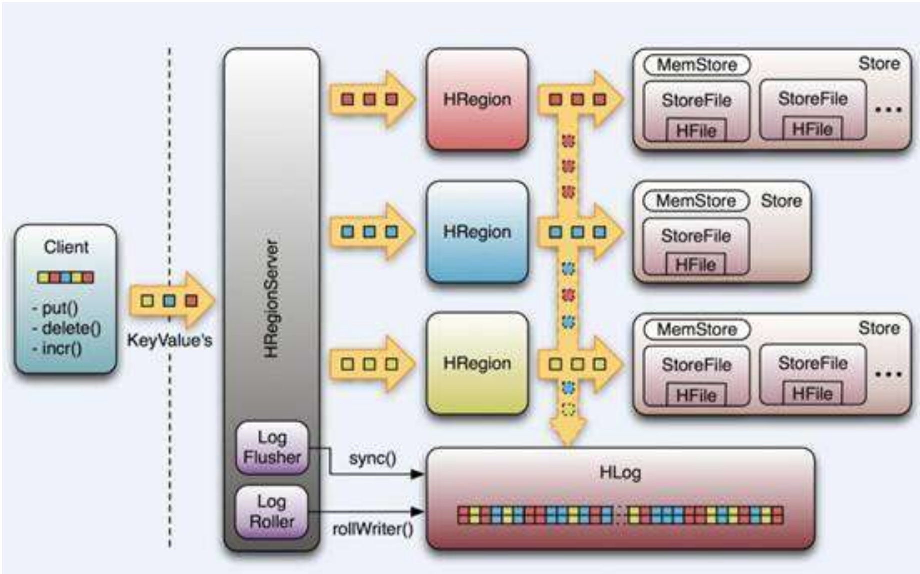


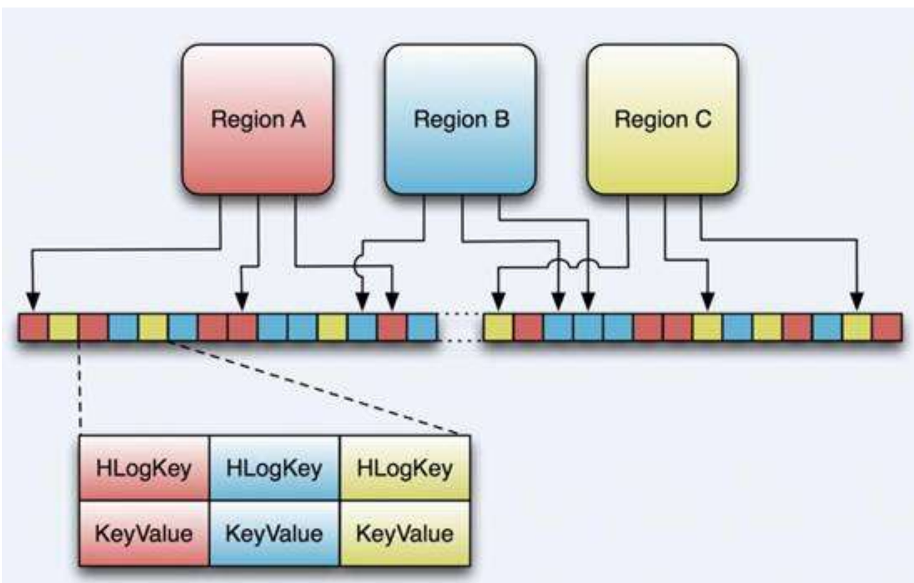
Similar to each other in outlook.

But very different in structure, Hbase span across multiple servers.



1. Data is written into MemStore and Hlog first.
2. After triggered, the data is flushed into StoreFile , which is the handler to HFile.
3. After StoreFile grows beyond a threshold, start compacting, merge StoreFiles .





Data is written into MemStore and HLog first.

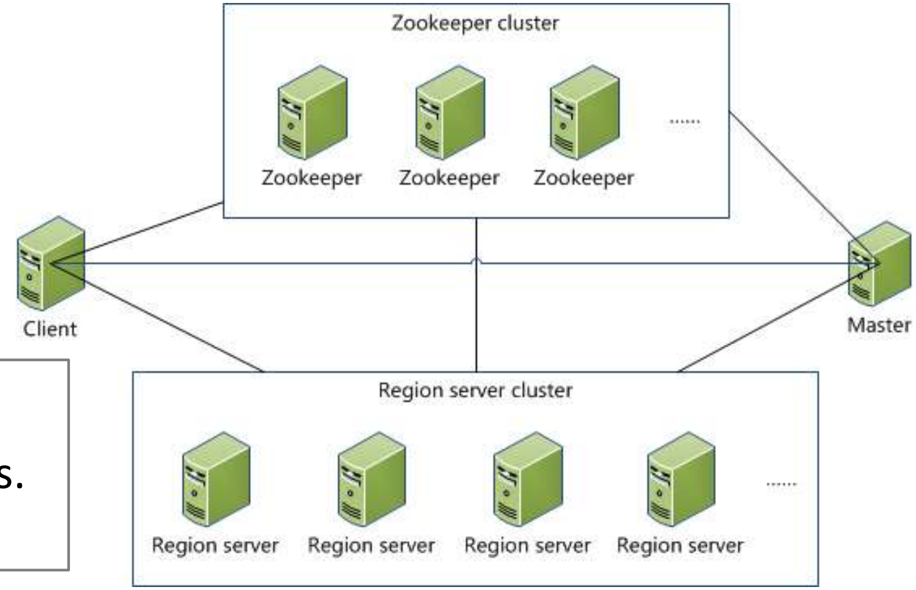
One RegionServer has only one HLog.

Various Regions' logs are multiplexed in the HLog.

When recovering RegionServer, we have to split the HLog, corresponding to different regions.

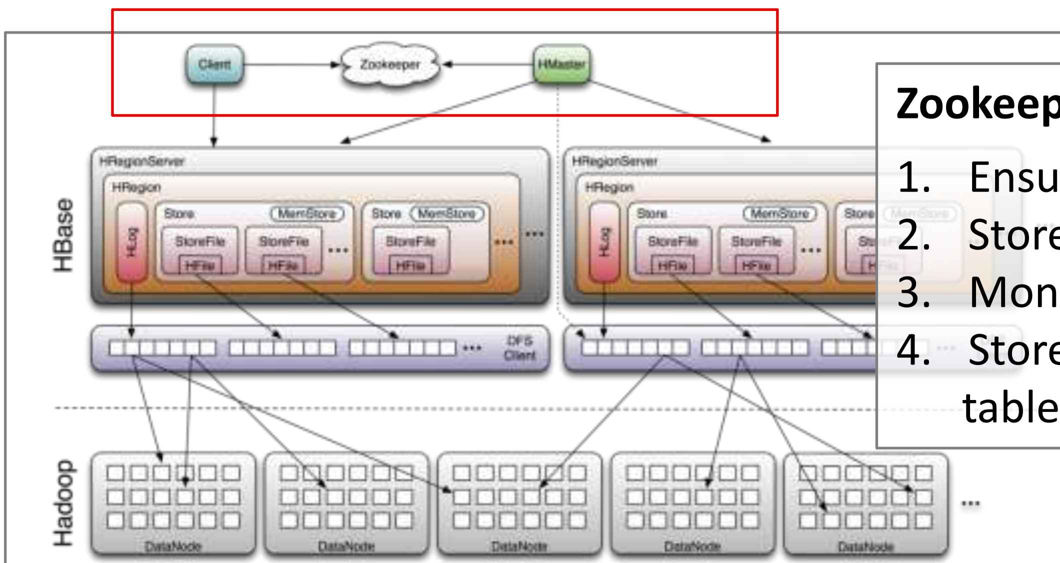
Client:

1. Gateway of HBase.
2. Cache the region positions.



Master:

1. Dispatch Regions to RegionServers.
2. Assign RegionServers.



Zookeeper:

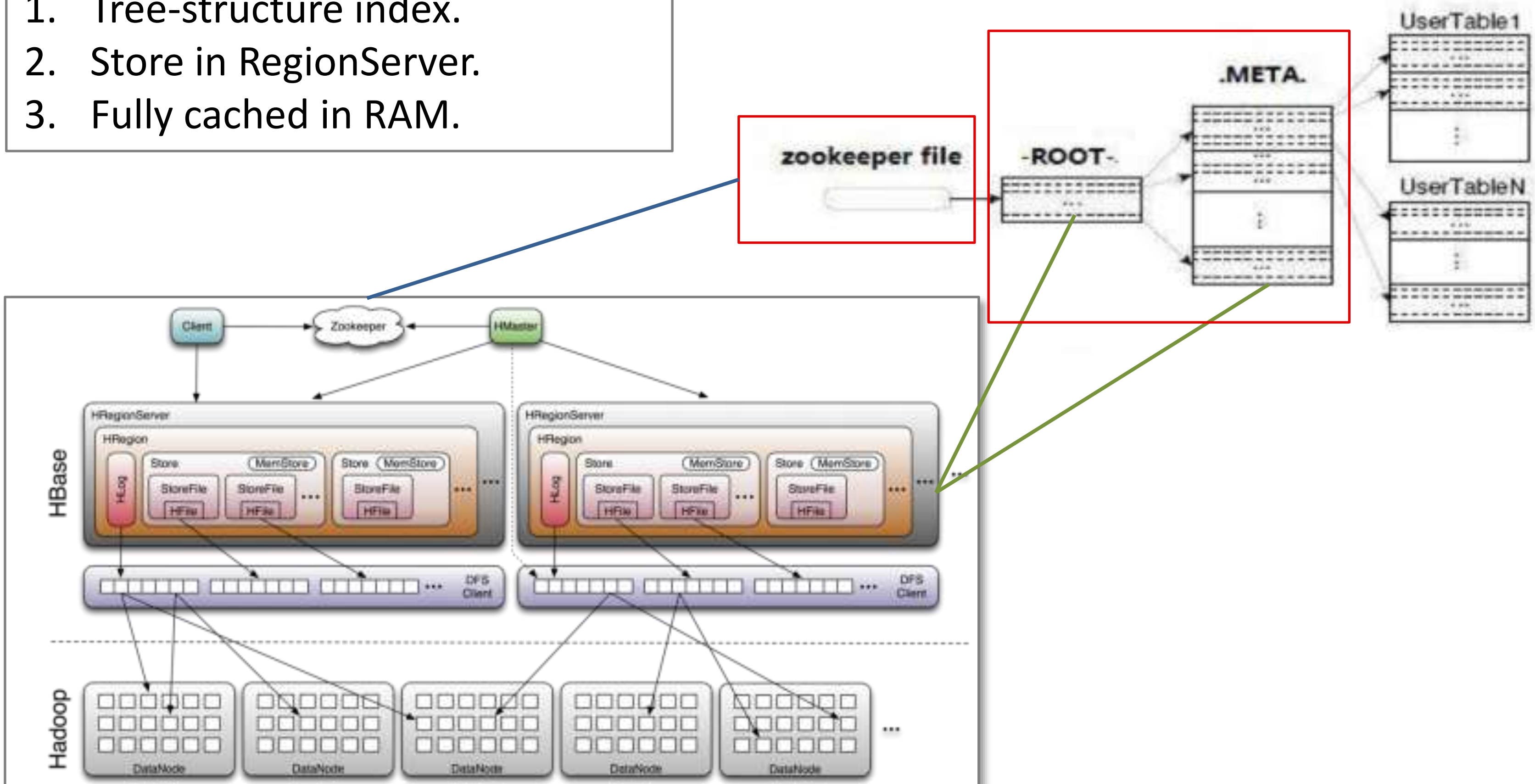
1. Ensure only one active master.
2. Store index –ROOT-.
3. Monitor RegionServer's aliveness.
4. Store the Hbase schema, i.e. table info, column family info…

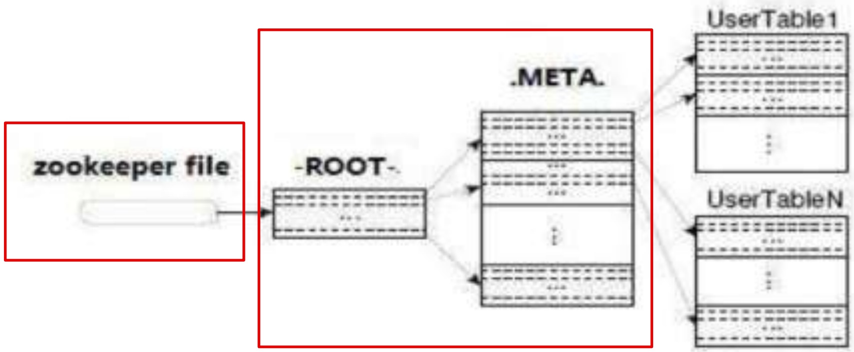
Zookeeper file:

Keep the pointer to the -ROOT- Region.

-ROOT- and .META.

1. Tree-structure index.
2. Store in RegionServer.
3. Fully cached in RAM.





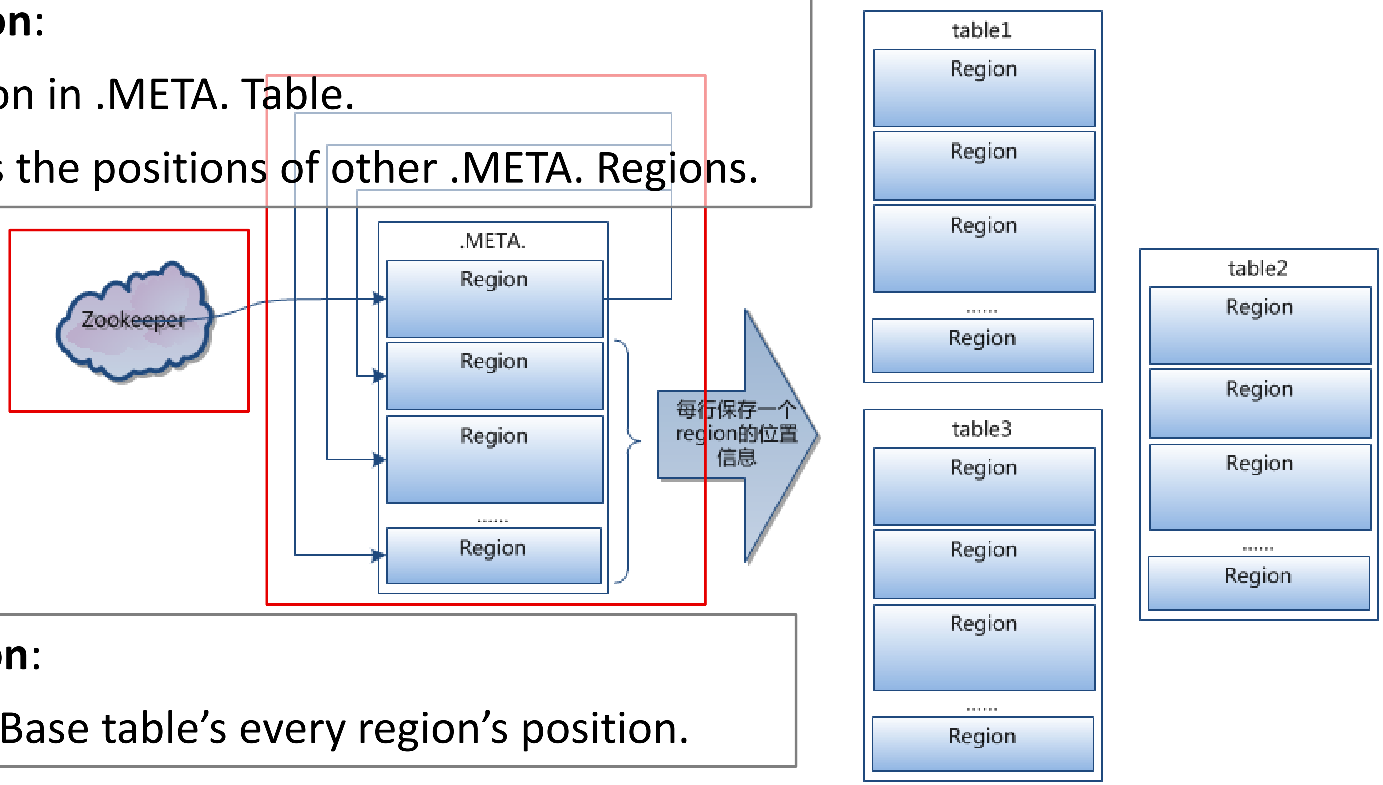
Zookeeper file:

Keep the position of the -ROOT- Region.

-ROOT- Region:

The first region in .META. Table.

-ROOT- keeps the positions of other .META. Regions.



.META. Region:

Keep every HBase table's every region's position.

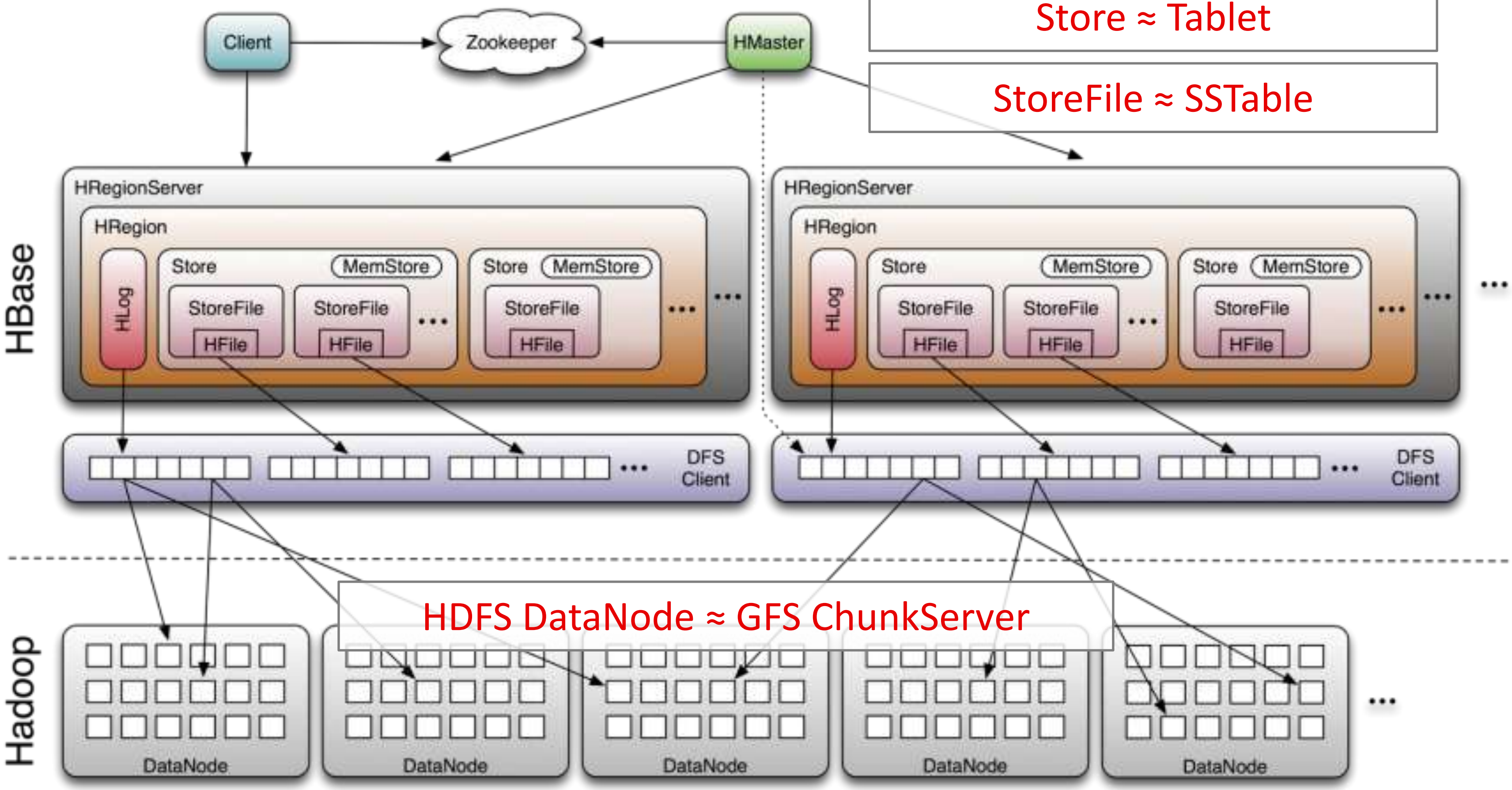
HBase 0.9 ≈ Google Bigtable 2006

Zookeeper ≈ Chubby

RegionServer ≈ TabletServer

Store ≈ Tablet

StoreFile ≈ SSTable



HDFS DataNode ≈ GFS ChunkServer

Benchmark tests of Bigtable 2006:

Each operation read/write 1 KB.

The number of ops per second, per tablet server.

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

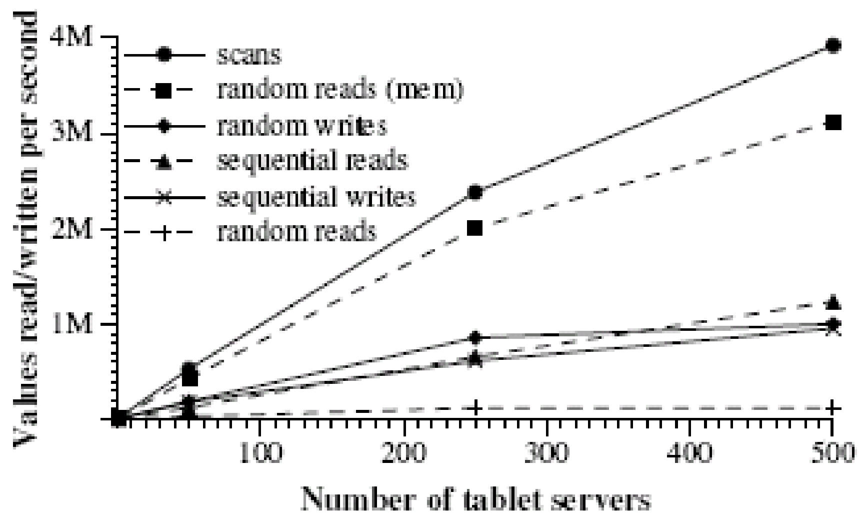
Aggregate number of ops per second:

Scan tables (sequentially) is fast.

Read from cache is fast.

Random read from disk is very slow.

Write to disk is slow.



- Hadoop HBase / Google Bigtable:

  Column-family based storage. Loosely-structured {key, value} data.

- One table can span onto multiple RegionServers / TabletServers:

  Each table may consist of multiple Stores / Tablets.

  Each Store / Tablet consists of multiple StoreFiles / SSTables.

  Each StoreFile / SSTablets consist of multiple DataBlocks.

  Each DataBlock consists of multiple ColumnFamilies.

- Data are written into cache and log first:

  Data are flushed from cache to file, then merge later,

  The logs are used for recovering.

- Tree-structure index:

  Zookeeper points to the -ROOT- Region,

  -ROOT- Region contains the positions of .META. Regions,

  .META. Region contains the positions of each region on each regioin-server.

Q&A

- No stupid questions, but it is stupid if not ask.
- When sleepy, the best trick to wake up is to ask questions.