HBase - Part 1 & 2

- Abhay Dandekar

CSV - Comma separated

- Comma separated
- Quote enclosed
    - 1, abc, 12/10/21
    - 2, “p,qr”, 19/08/21

TSV - Tab separated

Agenda

1. What is HBase?
2. Hbase v/s HDFS
3. HBase v/s RDBMS
4. ACID compliance
5. Data model Definitions
6. Entity Definitions
7. Hbase locking
8. HBase Architecture - Components and services
9. Hbase:meta
10. Client interactions

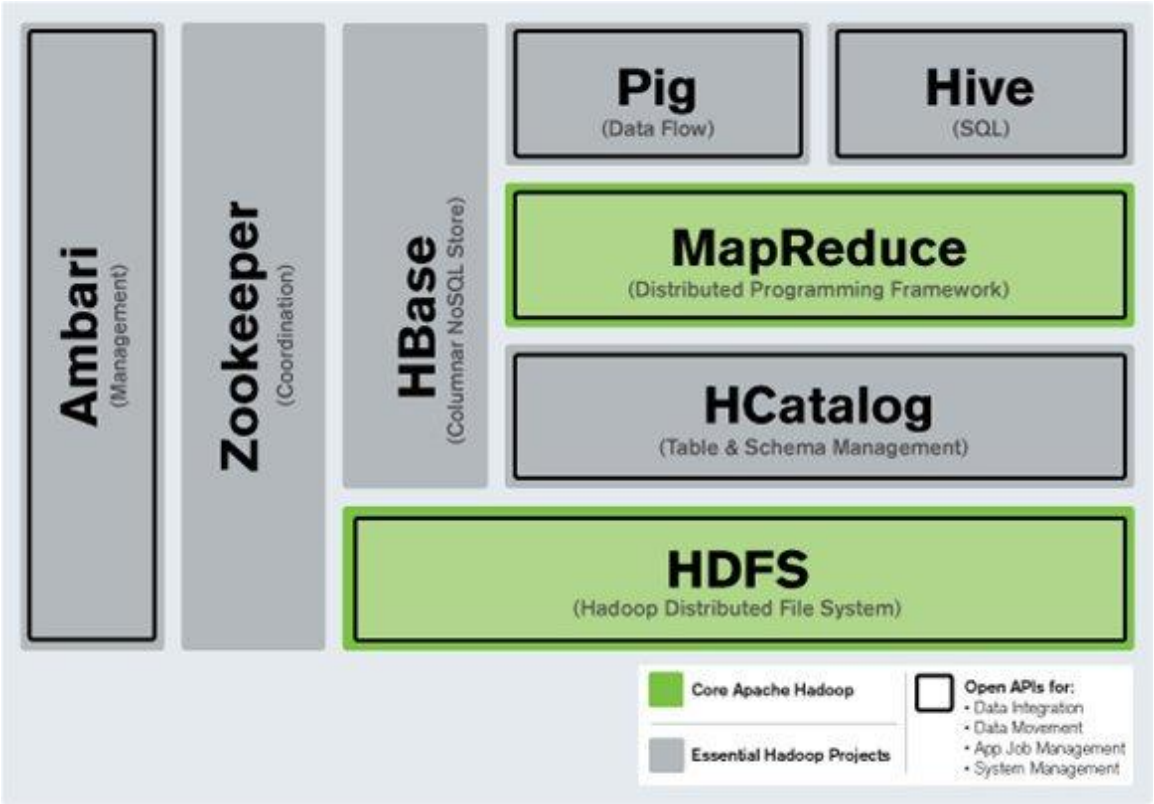
Agenda - continued

11. HBase Writes / Reads / Deletes
12. HBase Housekeeping
13. Region operations
14. HLog
15. Failovers
16. HBase entire picture
17. HBase access methods (Clients)
18. HBase CAP Theorem
19. Installation
20. Execution
21. To be continued …

What is HBase?

1. Modelled after Google's BigTable
2. BigTable is sparse, distributed, persistent multidimensional sorted map
3. Hbase is opensource implementation of Big Table
4. Distributed
5. Column oriented
6. NoSQL (i.e Non relational)
7. Database
8. Works on top of HDFS
9. Application which provides real-time and random reads/writes to very large datasets
10. Contributed by Facebook, Cloudera, Hortonworks etc

What is HBase?



Abhay Dandekar

HBase v/s HDFS

Feature	**HDFS**	**HBase**
Hadoop eco-system	Core part of Ecosystem	Not a core part of ecosystem, Built on HDFS
Access	Java API / HDFS commands	NoSQL interface
Reads	Full file scan	Random read
Writes	Append only	Random writes, Bulk loads
Structured storage	User defined, Avro or Sequential files	Sparse column family
Max Data size	30+ PB	~ 1 PB

HBase v/s RDBMS

Feature	**RDBMS**	**HBase**
Data structure	Row oriented	Column oriented
Transaction	Multi-Row ACID	Single row only
Query Language	SQL	NoSQL
Security	Authentication	HDFS + Auth
Indexed	On any column	Only on one column (row-key)
Max data size	TBs	~ 1 PB
Throughput	1k queries / sec	Million queries / sec

HBase ACID properties

1. HBase is not ACID compliant, because it does not support Isolation
2. A : Atomic
    - HBase guarantees updates for single record atomically
3. C : Consistent
    - HBase is inconsistent stage - No PK / FK relations
4. I : Isolation
    - HBase does not guarantee changes sequentially
5. D : Durable
    - HBase changes are durable in HDFS

Abhay Dandekar

Hbase Table - A view

HBase Table						
	ColumnFamily1		ColumnFamily2			ColumnFamily3
RowKey	Col1	Col2	Col3	Col4	Col5	Col6

Abhay Dandekar

Data Model - Definitions

1. Application stores data into labelled tables
2. Table - Tables are nothing but rows and columns
3. Table cells - intersection of rows and columns, timestamp versioned
4. RowKey - Each row has a key column called a RowKey
5. Datatype of RowKey can be anything, long, string, serialized data structures
6. Table rows are sorted by RowKey, a.k.a tables' primary key

Data Model - Definitions (part 2)

7. Row Columns are grouped into Column Families.
8. All members of column family start with a common prefix
9. Column family and column qualifier are separated by colon (:)
10. Columns families must be specified up front
11. New column family members can be added on demand.
12. Physically, all column family members are stored together

Entity definitions - Regions

1. Tables are partitioned horizontally
2. These partitions are nothing but Regions
3. Each Region comprises of a subset of table’s rows
4. Regions are the units that get distributed over the HBase cluster
5. Regions are created as the data gets inserted into the table

HBase locking

1. Row updates are atomic
2. There can be one or many columns being updated in a row transaction.
3. Maintained by managing the timestamp

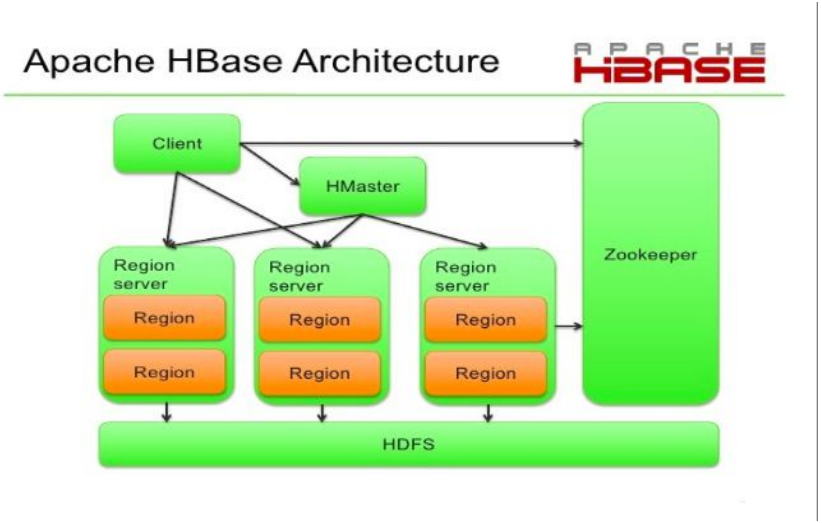
Note : Quick look at the following links:

http://blog.cloudera.com/blog/2012/06/hbase-io-hfile-input-output/

http://blog.cloudera.com/blog/2012/06/hbase-write-path/

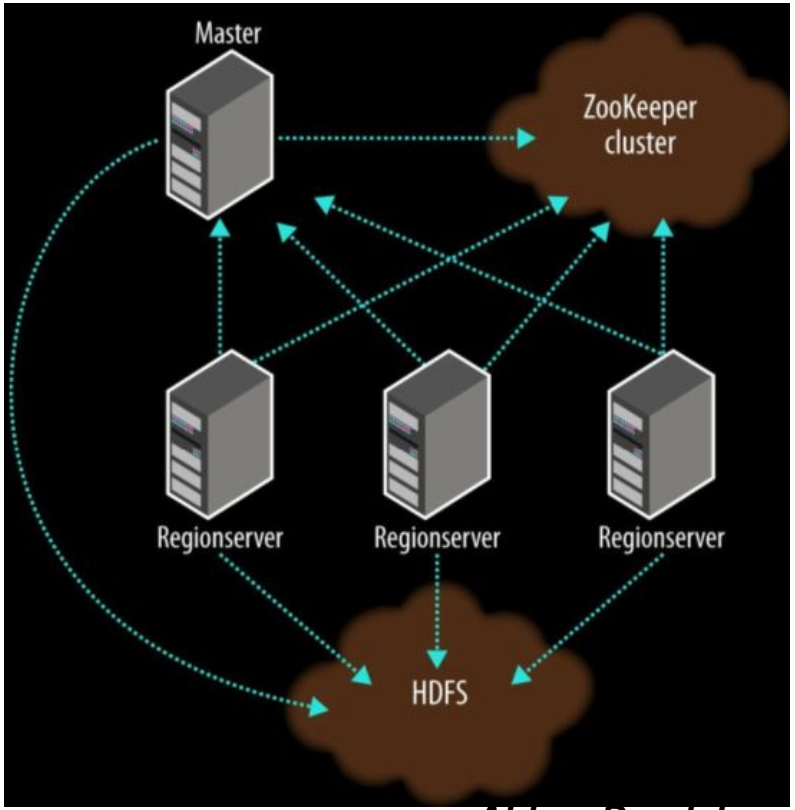
HBase Architecture - Components

1. HBase Master
   a. Orchestrates the cluster of region servers
   b. Assigning regions to region server
   c. Recovering from Region Server failures
2. Region Servers
   a. Carry regions
   b. Provide client's read/write request
   c. Manage region splits
3. Zookeeper
   a. Maintains address of cluster master
   b. Manages hbase:meta



HBase Architecture

1. Region Server list
    a. conf/regionservers
2. Site specific config
    a. conf/hbase-site.xml



Abhay Dandekar

hbase:meta

1. HBase maintains hbase:meta
   a. Current list of regions
   b. States
   c. Locations
2. Entries in hbase:meta are keyed by region name
3. Region name comprises of
   a. Table name
   b. Region's start row
   c. Creation TS
   d. Md5 sum of all of above 3 entities
4. Example:
   a. TestTable,xyz,1279729913622,1b6e176fb8d8aa88fd4ab6bc80247ece.

Abhay Dandekar

How Clients talk to HBase

1. New clients connect to ZooKeeper
2. Learns the location of hbase:meta
3. They retrieve hbase:meta
4. Looks up on hbase:meta to get table regions and its location
5. Thereafter, all interactions are with client and respective region servers
6. Clients caches hbase:meta and uses it to retrieve table regions
7. If there is a cache fault, it performs again from step 1.
8. Faults may happen in case of region movements, rebalancing etc

Writes in HBase

1. Writes are performed directly on region servers
2. Incoming writes are first appended to a Write-Ahead-Log (WAL)
3. Commit logs are written onto HDFS
4. Then they are appended to a in-memory memstore
5. When the memstore fills, data is flushed onto HDFS creating a HFile

Abhay Dandekar

Reads in HBase

1. Region’s memstore are consulted first
2. If memstore has the requested data, query completes
3. If not, all the underlying HFiles are read, in reverse order of HFiles i.e, latest HFile will be read first

Abhay Dandekar

Deletes in HBase

1. HBase client sends out a delete request
2. The record's "tombstone" marker will be enabled.
3. This method is also known as "Predicate deletion"

HBase Housekeeping

1. All the updates / deletes are written into the HFile
2. Deletes are basically adding delete markers.
3. With increasing number of files, the read-operation becomes slow
4. Hence, HBase compacts these files at regular intervals
5. This process is called “Compaction”
    a. Minor Compaction : It just merges HFiles
    b. Major Compaction : Removes the deleted records and merges the HFiles
6. All HFiles participating in “Compaction” must belong to same Column family.

Region Operations

1. Region Assignment
    a. HMaster does the region assignment
    b. Updates in hbase:meta
2. Region Split
    a. RegionServer splits the region, HMaster is not involved
    b. Occurs when the RegionServer accumulates a size threshold
    c. RegionServer will split the region into half.
    d. Split policy decides when to split a particular region. E.g SteppingSplitPolicy (default), BusyRegionSplitPolicy, ConstantSizeRegionSplitPolicy, DisabledRegionSplitPolicy, DelimitedKeyPrefixRegionSplitPolicy, and KeyPrefixRegionSplitPolicy
    e. DisabledRegionSplitPolicy blocks automatic region splitting. Reference: Link

Region operations - continues

3. Region Merge
    a. Merges two or more regions
    b. Regions must be from the same column family
    c. Regions may be moved across Region Servers for merging
    d. Region Server offlines the region to be merged
    e. Merges are performed on local
    f. Updates the hbase:meta
    g. Opens the newly merged region for query

HLog

1. Writes the edit logs to HDFS (a.k.a HStore)
2. There is one HLog per region server
3. All edits for all Regions carried by a particular RegionServer are entered first in the HLog.

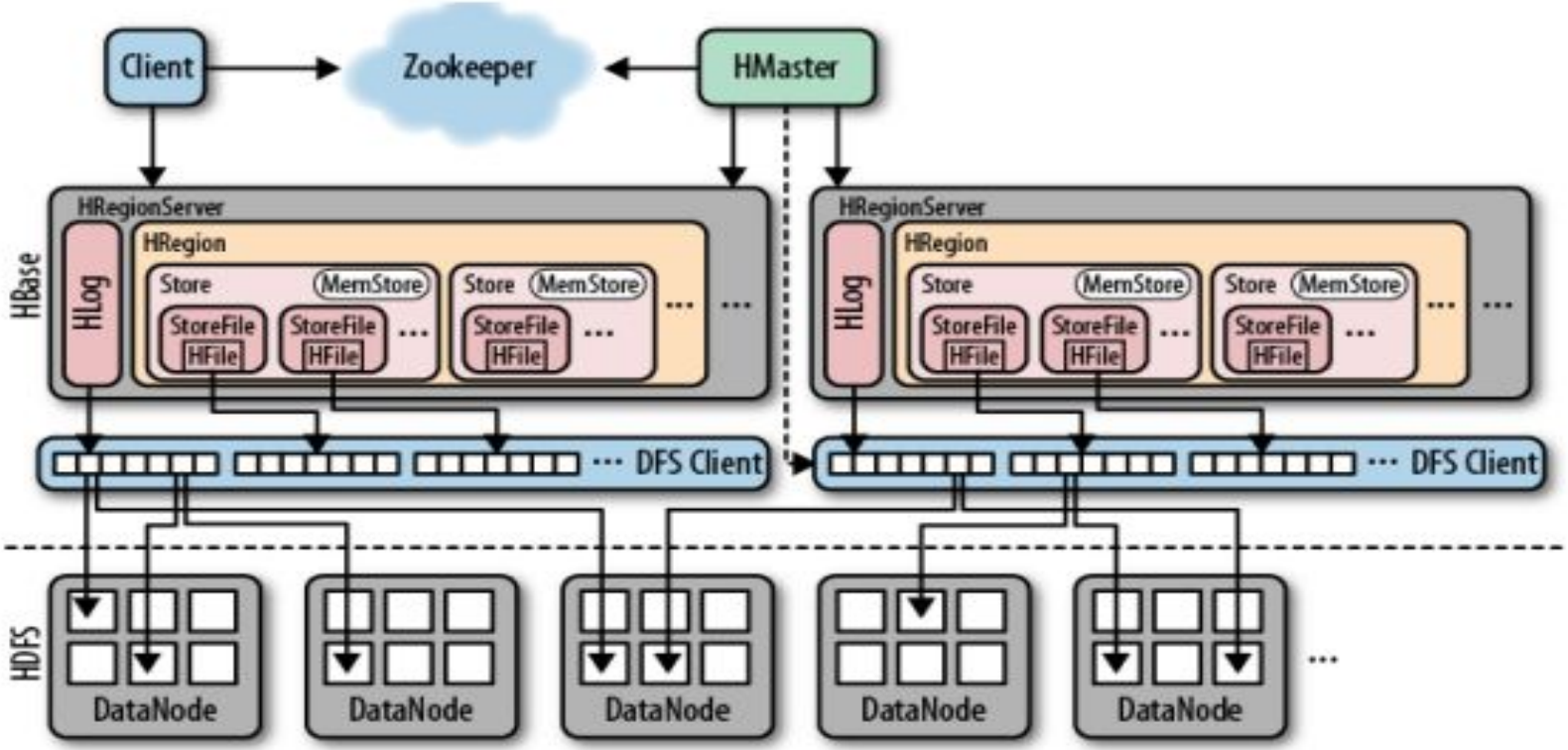
Failovers

1. RegionServer failovers
   a. HMaster will detect and re-assign the regions
2. HMaster failovers
   a. Similar to HDFS, we can have multi master HMaster
   b. One HMaster is active at a time

Since client caches hbase:meta from the RegionServers, and reads/writes directly to the RegionServer, HMaster failure may not hamper existing connections.

Abhay Dandekar

HBase - Entire picture



Abhay Dandekar

2 Transactions :

1. 1M inserts
2. 0.5M deletes / updates

Total records physically present in HBase = 1.5M

Logical non-deleted records [from a select count(*) query] = 0.5M

On major compaction, the total physical records = 0.5M

Hbase Access methods - (Clients)

1. HBase shell - ruby client with set of commands
2. Web interface - via browser, default port, 60010 (Ref : Cloudera Link)
3. Java API - Native APIs for HBase
4. REST API - REST APIs working on port 8080
   a. $ hbase rest start -p <PORT_NO>
5. Thrift - Enable access from any other language
6. Hive / Pig for analytics

HBase CAP Theorem

1. C : Consistency :
    - HBase is consistent across multi node reads
    - This is because of HDFS
2. A : Availability :
    - HBase does not guarantee 100% uptime i.e each request may not get response
3. P : Partition Tolerance
    - HBase is tolerant over partitions

Execution …

Ref : http://hbase.apache.org/book.html#shell

Installation

1. Untar hbase
    a. $ tar xvzf hbase-*.tgz
2. Export HBASE_HOME=<YOUR_UNTAR_LOCATION>
3. $ cd $HBASE_HOME
4. $ vi conf/hbase-site.xml
5. Enter the following properties
    a. hbase.rootdir,
    b. hbase.zookeeper.property.dataDir
    c. Hbase.regionserver.wal.codec
        i. See attached : hbase-site.xml
6. $ start-hbase.sh

Installation modes

1. Standalone
    - This is the default mode
    - It starts all processes in a single JVM
2. Pseudo distributed
    - Similar to HDFS pseudo distributed mode
    - It starts all processes in separate JVM but on a single machine
3. Distributed
    - Uses distributed nodes for starting various daemon processes.

To be continued …

Data types in HBase

1. It is all Byte Array
2. Put converts the data into Byte Array

Joins

1. HBase does not support joins
2. Joins have to be created programatically
3. Can use multi-table scanners. Link

Abhay Dandekar

Programmatic flow for Hbase client

1. Create the configuration
2. Retrieve the "*HBaseAdmin*" object
3. Using this HBaseAdmin object, we can perform following operations
    a. Create table
    b. List
    c. etc
4. To operate on Tables, we need to create a HTable object
    a. *HTable table = new HTable(config, tableName);*
5. To put data, use HTable.put()
6. To get data, user HTable.get()

Java client Execution …

Ref : http://hbase.apache.org/book.html#shell

Python lovers …

1. Can use Thrift … Link
2. Can use HappyBase … Link

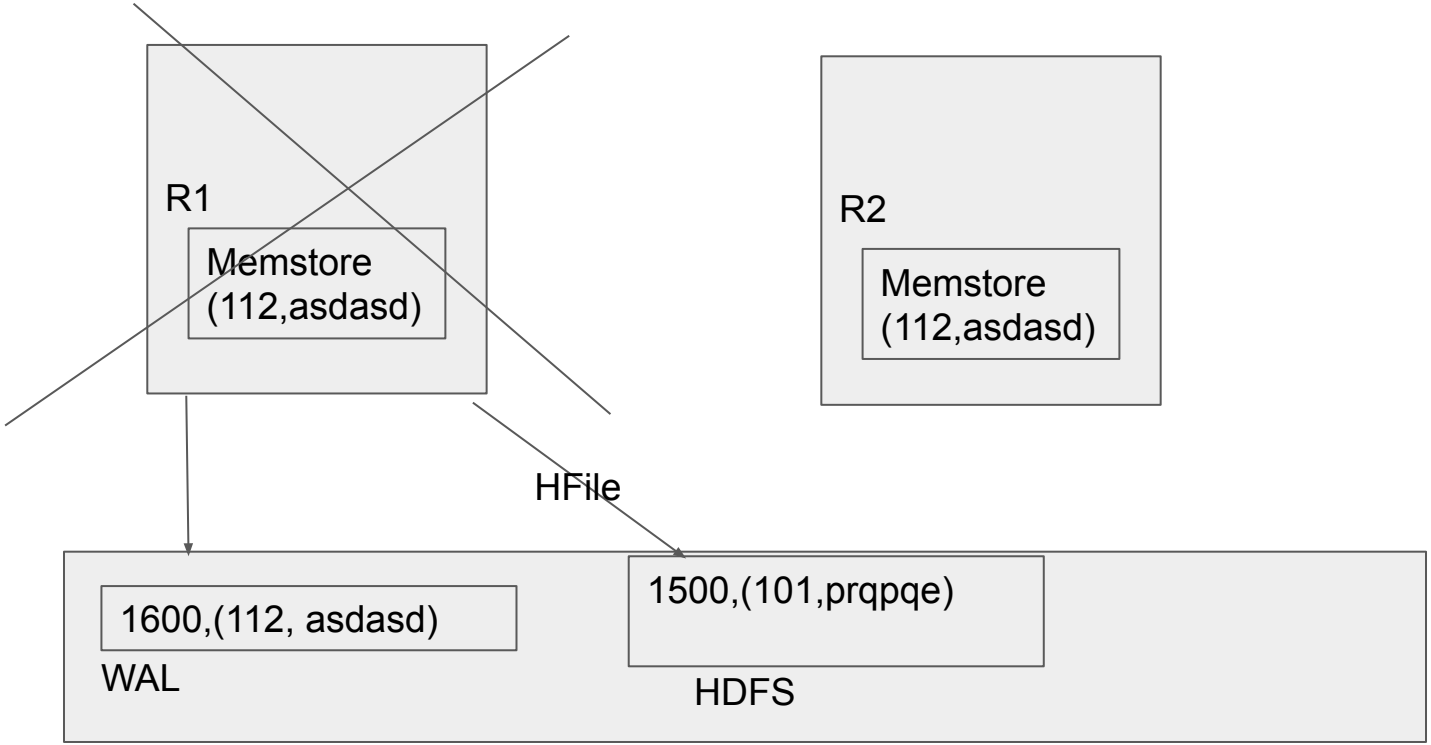
Abhay Dandekar

We are finally done !!!
Thank you

Abhay Dandekar

Appendix

Failure management in HBase Region (Region movement on failure)



Abhay Dandekar