DADS 6002 / CI 7301
Big Data Analytics

Hbase

- Hive can perform SQL-based analysis on large structured datasets stored in HDFS.
- Hive utilizes HDFS and MapReduce to process data in batch mode.
- Because of WORM properties of HDFS, Hive is not designed to support random real-time low latency read/write access to data.
- Some applications may require many possible structural variations in data records, e.g. different number of columns in rows of a table.

Hbase

- NoSQL is a broad term that refers to non-relational database and encompasses a wide collection of data storage models, including graph databases, document databases, Key/Value data stores and column-family databases.
- Hbase is one of NoSQL that is classified as a column-family or column oriented database.

Hbase

- It is modeled on Google’s BigTable architecture which provides :

1. Random (row-level) read/write access
2. Strong Consistency
3. Schema-less or flexible data modeling

- Hbase organizes data into tables that contain rows.
- Within a table, rows are identified by their unique row key.

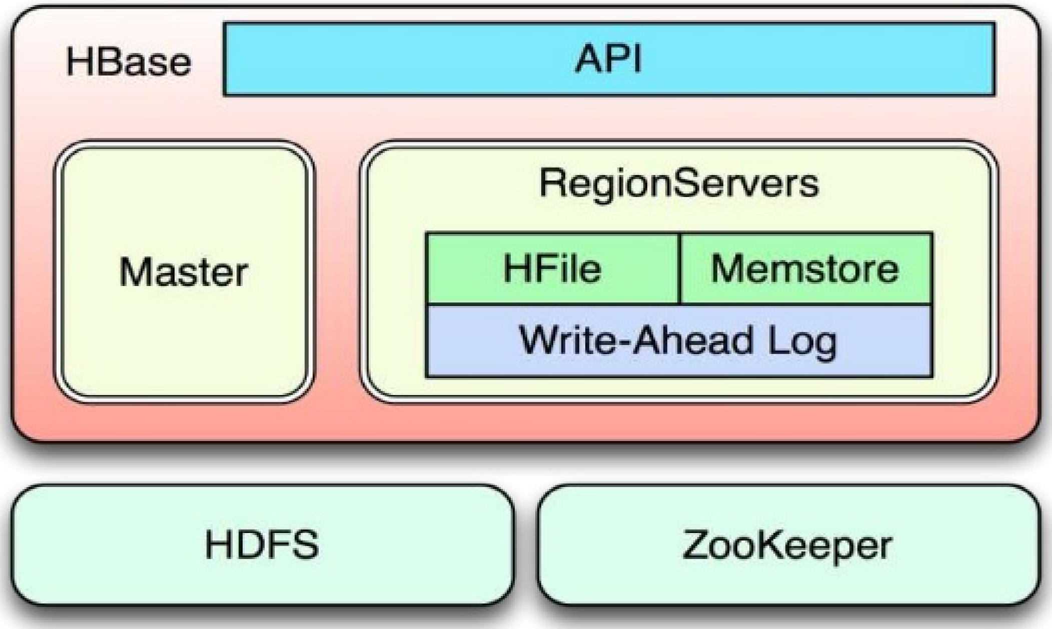
Hbase

- Row keys do not have a data type and stored as byte arrays.
- Hence, row keys can be strings to binary representations of longs or serialized data structures.
- Table rows are sorted by their row keys which are automatically indexed.
- Data within a row is grouped into column families, which consist of related columns.

Hbase

- A table is divided into regions (sets of rows) which are managed by region servers.
- Data in a region is stored in HFiles (in a form of Key/Value pairs) on HDFS.

HBase Architecture



Hbase

- Master responsible
  - Coordinating the region servers

    Assigning regions to region servers, re-assigning regions for recovery or load balancing

    Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)
  - Admin functions

    Interface for creating, deleting, updating tables

Hbase

- A Region Server runs on an HDFS data node and has the following components:

1. WAL: Write Ahead Log is a file on the Hadoop distributed file system (HDFS). The WAL is used to store new data (in Memstore) that hasn't yet been persisted to permanent storage (HFiles) ; it is used for recovery in the case of region server failure (replay the updates to restore data)
2. BlockCache: is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when Full.

Hbase

3. Memstore: is the write cache. It stores or buffers new data which has not yet been written to disk (before writing to disk). It is sorted before writing to disk. There is one MemStore per column family per region.

   When the MemStore accumulates enough data, the entire sorted set is written to a new HFile in HDFS. HBase uses multiple HFiles per column family.

4. HFiles store the rows as sorted KeyValues on disk.

Hbase

- For every read on a table, BlockCache and Memstore are searched first for the given rowkey. If not found, then the Hfiles of the table are searched. Minor compaction is performed automatically to merge some small Hfiles into one Hfile to improve read performance in the future.
- Major_compaction command can be executed manually to merge all Hfiles for a table into one Hfile.

Hbase

- ZooKeeper , a coordinator

  - HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Servers send heartbeats (messages) regularly to ZooKeeper if they are still active.

  - HBase Catalog table called the META table, which holds the locations of all regions along with their starting rowkeys. ZooKeeper keeps the location of the region server where META table is located. To look up for a particular region for a given rowkey, clients need to ask for the location of the region server that manages the META table from ZooKeeper, before proceeds to look up for the region containing a row for a given rowkey from the META table.

Hbase

Example

- Person Table
- Row key : Person ID
- Column-Family : personal-data
    - Column : name
    - Column : address
- Column-Family : Demographic
    - Column : birthdate
    - Column : gender
- Column Family : …….

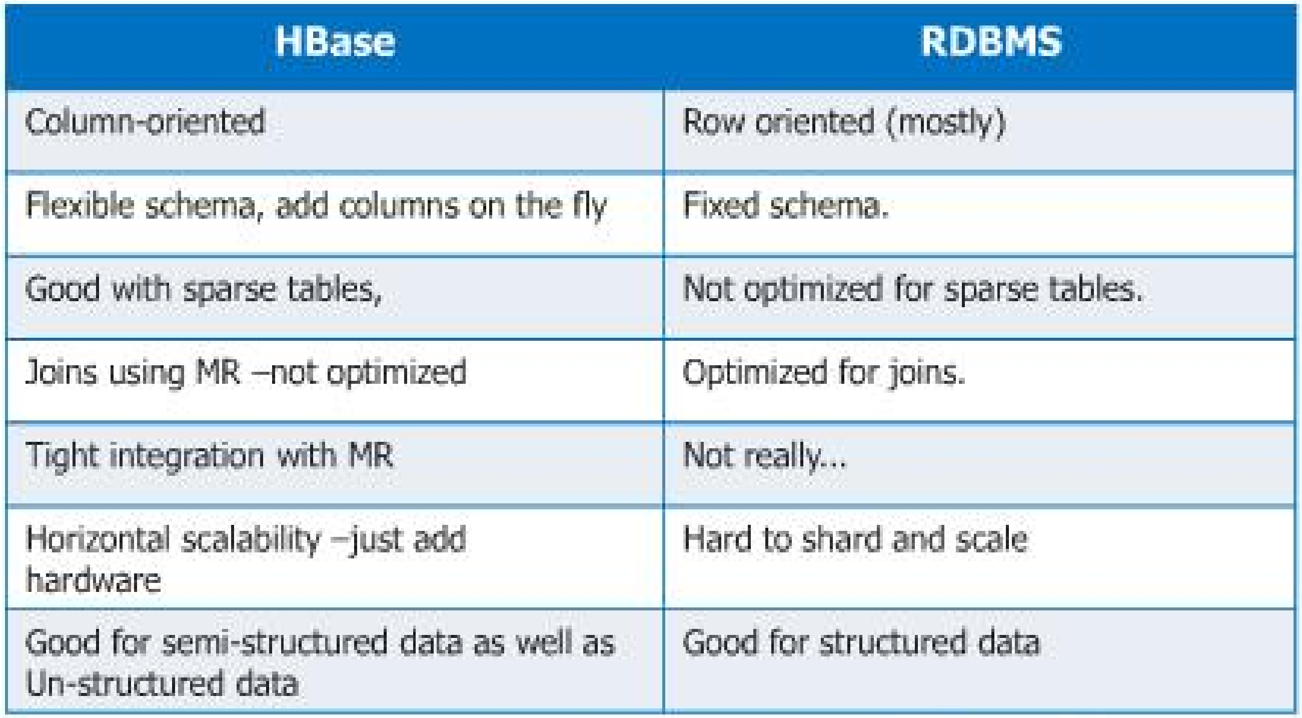
Hbase

- Advantage of storing data in columns rather than rows is that aggregation on columns can be computed over large sets of data where not all column values of rows are presented.
- Column families need to be defined up front before we can begin inserting data into a particular row and column.
- Table cell is the intersection of row and column coordinates. Each cell is versioned by a timestamp, store as a long integer representing milliseconds since Jan 1, 1970 UTC

Hbase

- Time dimension is indexed in decreasing order, so that when reading from Hbase, the most recent values are found first.
- The contents of a cell can be referenced by a { rowkey, column, timestamp }
- Hbase does not support join operations and provides only a single indexed rowkey.
- Hbase allows dynamic column definition at runtime. Hence, column families and column can be inserted/deleted dynamically.

Hbase



HBase	RDBMS
Column-oriented	Row oriented (mostly)
Flexible schema, add columns on the fly	Fixed schema.
Good with sparse tables,	Not optimized for sparse tables.
Joins using MR –not optimized	Optimized for joins.
Tight integration with MR	Not really...
Horizontal scalability –just add hardware	Hard to shard and scale
Good for semi-structured data as well as Un-structured data	Good for structured data

Hbase

- To start Hbase CLI

```
# hbase shell
hbase>
```

- To exit Hbase CLI

```
hbase> quit
```

Hbase

- To create a table, a table name and at least one column family name must be specified. Namespace (database name) can be declared. If not, default namespace will be used.

  ```
  hbase> create ‘linkshare’, ’link’
  ```

- A single table called linkshare is created in the default namespace with one column family named link.
- hbase> list

Hbase

- To alter the table structure after creation, e.g. changing or adding column families, we need to first disable the table so that clients will not be able to access the table during the alter operation.
- Hbase> disable ‘linkshare’
- Hbase> alter ‘linkshare’, ’statistics’
- Hbase> enable ‘linkshare’
- Hbase> describe ‘linkshare’

Hbase

- By default Hbase stores rows in sorted order by row key so that similar keys are sorted to the same region server.
- While this enables faster range scans, it can lead to uneven load on particular servers during read/write operation, i.e. some keys may be frequently accessed and become hotspots.

Hbase

- Inserting data with put

  hbase > put 'linkshare', 'org.hbase.www', 'link:title', 'Apache Hbase'
- A counter is associated with each column of a row. To increment the counter

  hbase > incr 'linkshare', 'org.hbase.www', 'statistics:share', 1
- To get the value in the counter

  hbase > get_counter 'linkshare', 'org.hbase.www', 'statistics: share'

Hbase

- Get row or cell values

  hbase > get ‘linkshare’,’org.hbase.www’, { COLUMN => ‘link:title’ }

  hbase > get ‘linkshare’,’org.hbase.www’, { COLUMN => [‘link:title’, ‘statistics:share’] }

- or (for short)

  hbase > get ‘linkshare’,’org.hbase.www’, ‘link:title’

  hbase > get ‘linkshare’,’org.hbase.www’, ‘link:title’, ‘statistics:share’

Hbase

- Specify a time range of values of data to be retrieved.

  hbase > get ‘linkshare’, ‘org.hbase.www’, ‘link:title’, { TIMERANGE => [1399887705673, 1400133976734] }
- Start and end timestamps in milliseconds
- Instead of explicit timestamp ranges, we can specify a certain number of versions of data to be retrieved.

Hbase

hbase > get ‘linkshare’, ‘org.hbase.www’, { COLUMN => ‘link:title’, VERSIONS => 2 }

The two previous version of the specified cells will be retrieved. By default, maximum number of versions to be stored for a column is 1. Version 1 is the latest one.

To change the max number of versions of a column family

hbase > alter ‘linkshare’, { NAME => ‘link’, VERSIONS => 5 }

Hbase

- Scan rows iterate through row data to match against the source specifications.
- Entire Hbase table or a specified range of rows can be scanned.

  hbase> scan ‘linkshare’
- Scan the entire table. Rows in Hbase are stored in lexicographical order. For example, numbers from 1 to 100 will be ordered as follows:

  1, 10, 100, 11, 12, 13, …, 18, 19, 2, 20, 21, 23, …, 9, 90, 91, …., 98, 99

Hbase

```
Hbase> scan ‘linkshare’, { COLUMN => ‘link:little’,
    STARTROW => ‘org.hbase.www’ }
```

```
Hbase> scan ‘linkshare’, { COLUMN => ‘link:little’,
    STARTROW => ‘org.hbase.www’, ENDROW =>
    ‘org.hive.www’ }
```

STARTROW and ENDROW values do not require an exact match. It will match the first row key that is equal to or larger than the given STARTROW and less than the ENDROW. If ENDROW is not specified, it scans till the end of the table.

Hbase

- Hbase provides a number of filter classes to filter the rows returned from a get or scan operation.
- Available filters include

1. RowFilter is used for filtering rows based on row key values. ValueFilter is used for filtering columns in each row based on their values.
2. ColumnRangeFilter is used to get a slice of the columns of very wide row (filtering columns in each row, for which their values are within a range).
3. SingleColumnValueFilter is used to filter rows based on a value of a specific column.
4. RegexStringComparator is used to test if a given regular expression matches a cell value in the column.

- Use show_filters command to display all available filters

Hbase

```
Hbase> scan ‘linkshare’, { FILTER => RowFilter
   (>, ‘binary:xyz) }

Hbase> scan ‘linkshare’, { COLUMN => ‘link:title’,
   FILTER => “ValueFilter ( <=, ‘binary:Apache’ )” }
```

Hbase

- Delete a cell in a row

  hbase > delete ‘linkshare’, ‘org.hbase.www’, ‘link:title’

- Update a cell in a row with a new value

  hbase > put ‘linkshare’, ‘org.hbase.www’, ‘link:title’, ‘new value’

  hbase > scan ‘linkshare’

Hbase

- Delete a table

  First disable the table then delete it using drop command

```
hbase> create ‘test’, ‘cf’
hbase> list
hbase> disable ‘test’
hbase> drop ‘test’
hbase> list
```

Hbase

- Flush memstore into Hfile

  hbase> flush ‘table_name’