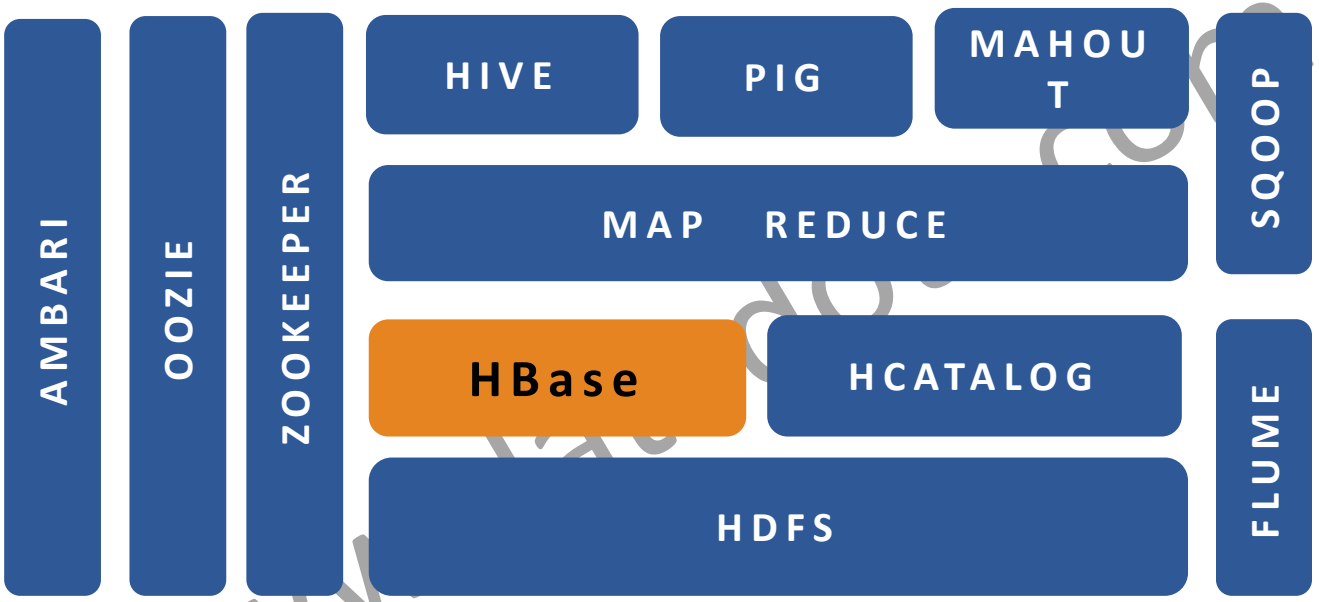
APACHE HBASE

Senthil

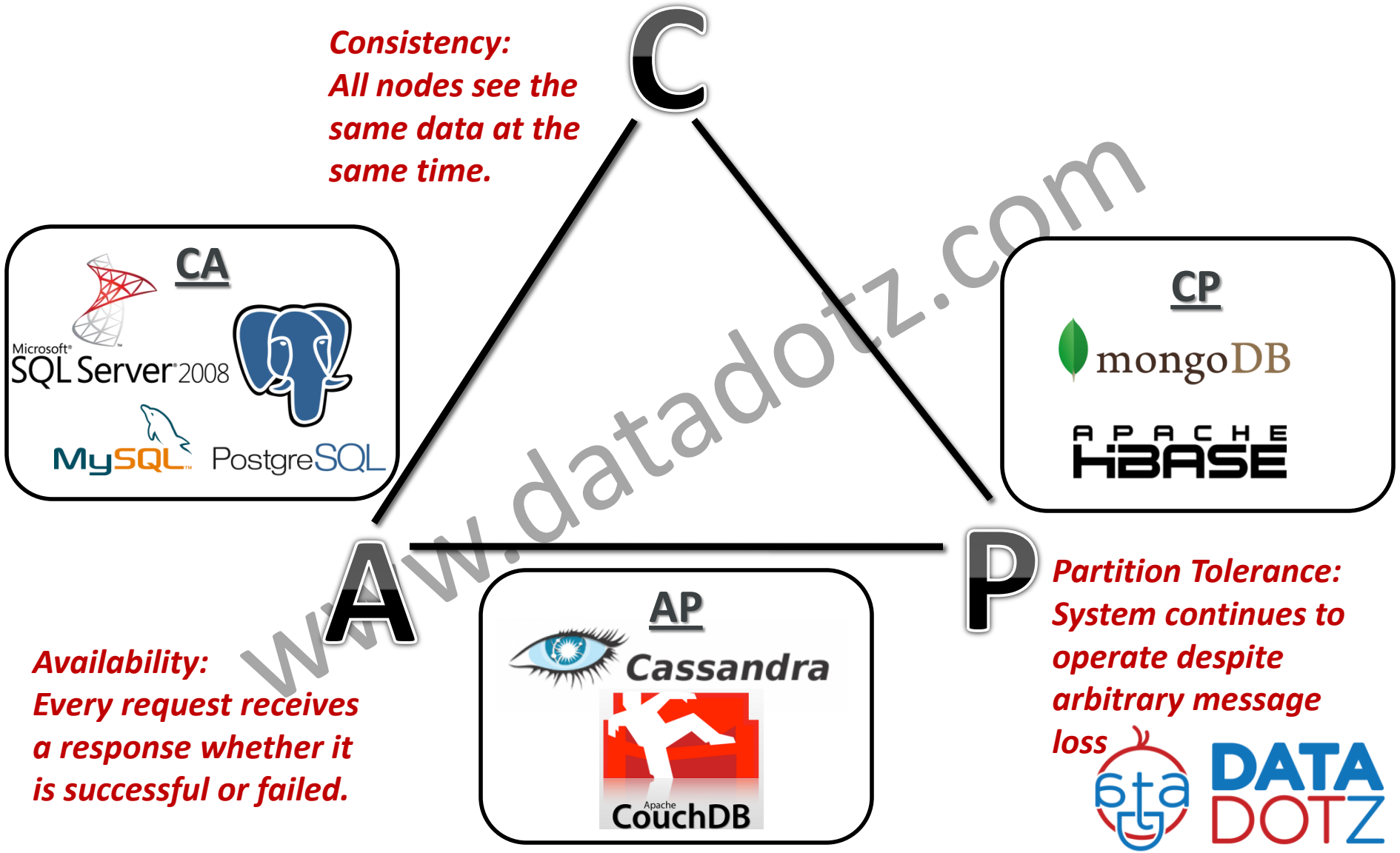
DATA DOTZ

Hbase in Hadoop Ecosystem



www.datadotz.com

CAP Theorem



www.datadotz.com

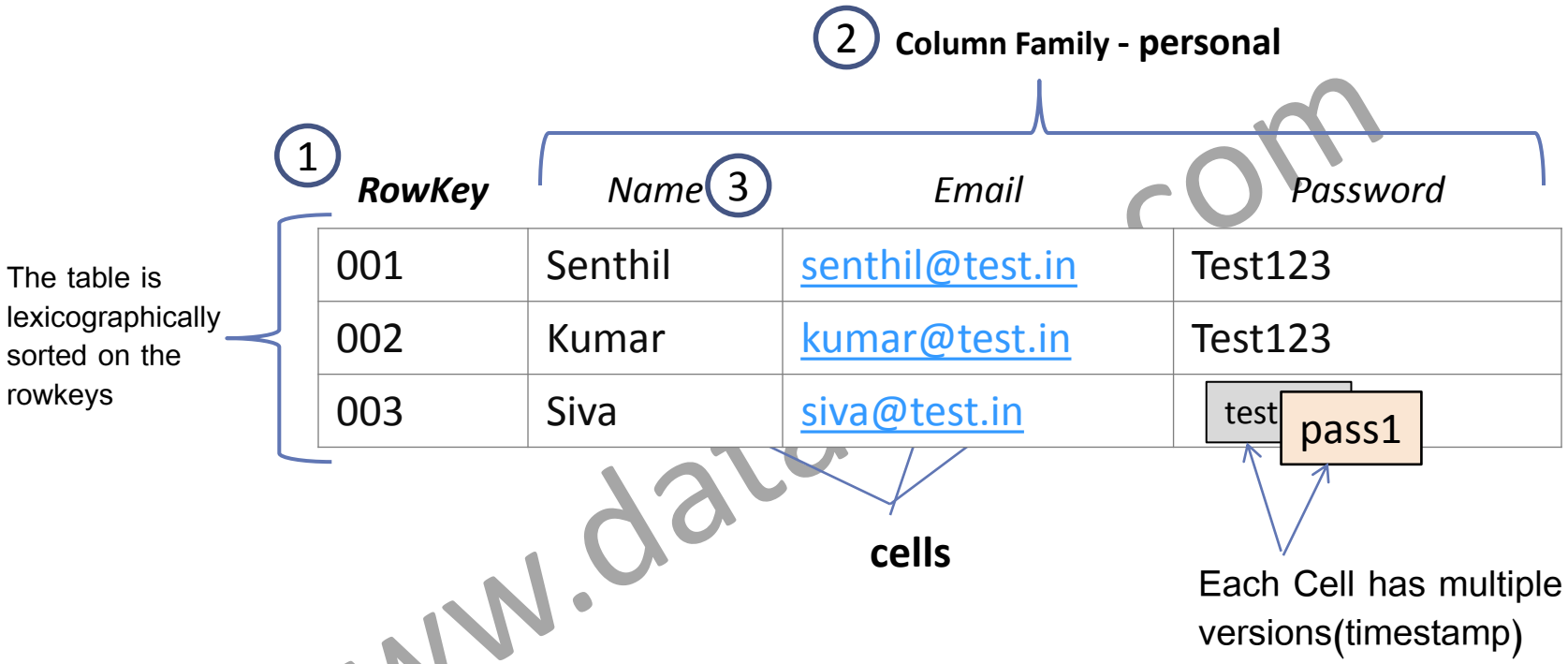
DATA DOTZ

HBASE is CP

- Brewer's **CAP theorem** For NoSQL datastores
  - Consistency
    - Eventual, Weak, Strong
  - Availability
  - Partition Tolerance

www.datadotz.com

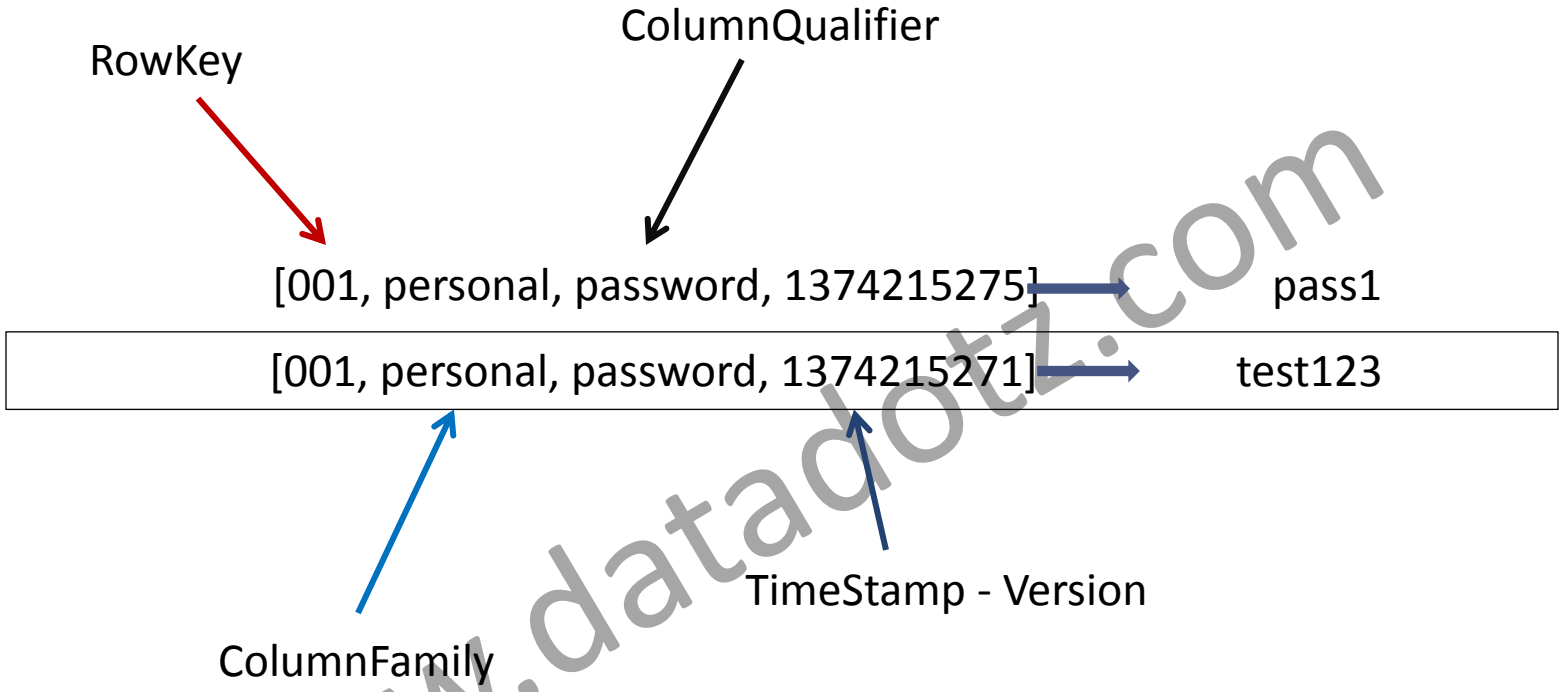
Tabular Format



www.datadotz.com

The coordinates used to identify data in an HBase table are:
(1) rowkey, (2) column family, (3) column qualifier, (4) version

Key Value Store



www.datadotz.com

HBase is an open source distributed, sparse, multidimensional sorted map modeled after Google's Big Table

Map -> Associates keys to values

Sorted -> Ordered by key (efficient look-ups)

Multidimensional -> Key is formed by several values

Persistent -> Once written, it is there until removed

Distributed -> Stored across different nodes

Sparse -> Many (most) values are not defined

www.datadotz.com

Usage Scenarios

- Lots of data
  - 100s of Gigabytes upto Petabytes
- High write throughput
  - 1000s /second (per second)
- Scalable cache capacity
  - Adding nodes adds to available cache
- Data layout
  - Excels at key lookup
  - No Penalty for sparse columns

www.datadotz.com

DATA DOTZ

Hbase vs RDBMS

Column-oriented	Row oriented (mostly)
Flexible schema, add columns on the fly	Fixed schema
Good with sparse tables	Not optimized for sparse tables
No query language	SQL
Wide tables	Narrow tables
Joins using MR –not optimized	Optimized for joins (small, fast ones too!)
Tight integration with MR	Not really...
De-normalize your data	Normalize as you can
Horizontal scalability –just add hardware	Hard to shard and scale
Consistent	Consistent
No transactions	Transactional
Good for semi-structured data as well as structured data	Good for structured data
No Secondary Index	Secondary Index Available

Hbase is built on Hadoop

- Hadoop provides
  - Fault tolerance
  - Scalability
  - Batch Processing with Map reduce

www.datadotz.com

DATA DOTZ

HDFS + HBase

- HDFS lacks Random read/write capabilities
- Hbase = HDFS + Random read/writes
- Hbase uses HDFS for storage but adds random read and write by appending writes to a logs and occasionally merging those with HDFS files

www.datadotz.com

DATA DOTZ

Data Model

- Every row has a row key (analogous to a primary key)
  - Rows are stored by row key for fast lookups
- A table may have 1 or more column families
  - Common to have a small number of column families
  - A column family can have number of columns
- Each row has a timestamp
  - Multiple versions of a row can exist

www.datadotz.com

DATA DOTZ

Column Family Attributes

- COMPRESSION
  - NONE.GZ,LZO
  - NONE
- VERSIONS
  - 1+
  - 3
- TTL
  - 1-2147483647
  - 2147483647
- BLOCKSIZE
  - 1byte – 2GB
  - 64k
- IN_MEMORY
  - True,false
  - false
- BLOCKCACHE
  - True.false
  - true

www.datadotz.com

DATA DOTZ

Prerequisites for installation

- JAVA_HOME
- ssh for password less login
- Loopback IP
  - Comment 127.0.1.1 in /etc/hosts file (**only for ubuntu machine**)
- HADOOP
  - Start the Hadoop Clsuter

www.datadotz.com

Installation

- Download recent version of hbase-X.XX.X.tar.gz from hbase.apche.org
- hbase-env.sh
  - export JAVA_HOME=/usr/jdk1.6.0_32 (Specify the path where java installed)
  - export HBASE_HEAPSIZE=1000
  - export HBASE_MANAGES_ZK=true
- hbase-site.xml
  - hbase.rootdir
  - hbase.zookeeper.property.dataDir
  - fs.default.name
  - hbase.zookeeper.quorum
- regionservers
- bin/start-hbase.sh

www.datadotz.com

Default Ports

- master
  - RPC - 60000
  - UI - 60010
- regionservers
  - RPC - 60020
  - UI - 60030

www.datadotz.com

DATA DOTZ

Hbase Shell

```
saravanan@saravana:~/hadoop/hbase-0.94.15$ bin/start-hbase.sh
localhost: starting zookeeper, logging to /home/saravanan/hadoop/hbase-0.94.15/bin/../logs/hbase-saravanan-zookeeper-saravana.out
starting master, logging to /home/saravanan/hadoop/hbase-0.94.15/bin/../logs/hbase-saravanan-master-saravana.out
localhost: starting regionserver, logging to /home/saravanan/hadoop/hbase-0.94.15/bin/../logs/hbase-saravanan-regionserver-saravana.out
saravanan@saravana:~/hadoop/hbase-0.94.15$ jps
5335 TaskTracker
11181 Jps
4767 NameNode
4928 DataNode
5182 JobTracker
10803 HMaster
11007 HRegionServer
5089 SecondaryNameNode
10725 HQuorumPeer
saravanan@saravana:~/hadoop/hbase-0.94.15$ bin/hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.15, r1551829, Wed Dec 18 22:35:57 UTC 2013

hbase(main):001:0> list
TABLE
0 row(s) in 0.5070 seconds
```

Shell commands

- help
  - Lists all the shell commands
- status
  - Shows basic status about the cluster
- list
  - Lists all user tables in HBase
- describe <table name>
  - Returns the structure of the table

www.datadotz.com

DATA DOTZ

Shell Commands - contd

- **Create** – Creating a table (‘Patient’) with two Column families (‘Personal’,’Medical’)

Command: **create ‘Patient’ , ’Personal’ , ’Medical’**

```
hbase(main):002:0> create 'Patient','Personal','Medical'
0 row(s) in 1.2140 seconds

hbase(main):003:0>
```

www.datadotz.com

DATA DOTZ

Shell Commands - contd

- **Put**– Inserting a record(Patient name) to table (patient) under column family (Personal)

- Command: **put 'Patient' , '001' , 'Personal:pname' ,'datadotz'**

```
hbase(main):003:0> put 'Patient','001','Personal:pname','datadotz'
0 row(s) in 0.0830 seconds

hbase(main):004:0>
```

www.datadotz.com

Shell Commands - contd

- **Scan**– Retrieving all Records from the table (patient)

- Command: **scan 'Patient'**

```
hbase(main):013:0> scan 'Patient'
ROW                           COLUMN+CELL
 001                          column=Personal:pname, timestamp=1406299157626, value=datadotz
 002                          column=Personal:pname, timestamp=1406299252371, value=saravanan
 003                          column=Personal:pname, timestamp=1406299260881, value=gowtham
 004                          column=Personal:pname, timestamp=1406299268535, value=amudhan
 005                          column=Personal:pname, timestamp=1406299275355, value=alex
5 row(s) in 0.0240 seconds
```

www.datadotz.com

Shell Commands - contd

- **Get**– Retrieving a specific Records (Patient ID 001) from table (patient)

- Command: **get 'Patient' , '004'**

```
hbase(main):016:0> get 'Patient','002'
COLUMN                          CELL
 Personal:age                   timestamp=1406299437183, value=24
 Personal:pname                 timestamp=1406299252371, value=saravanan
2 row(s) in 0.0190 seconds
```

www.datadotz.com

Shell Commands - contd

- Retrieving more VERSIONS

- Command: **scan 'Patient',{VERSIONS => 3}**

```
hbase(main):018:0> scan 'Patient', {VERSIONS => 3}
ROW                                COLUMN+CELL
 001                               column=Personal:pname, timestamp=1406299157626, value=datadotz
 002                               column=Personal:age, timestamp=14062994[illegible]
 002                               column=Personal:pname, timestamp=1406299252371, value=saravanan
 002                               column=Personal:pname, timestamp=1406299204547, value=gowtham
 002                               column=Personal:pname, timestamp=1406299197525, value=saravanan
 003                               column=Personal:pname, timestamp=1406299260881, value=gowtham
 003                               column=Personal:pname, timestamp=1406299217789, value=amudhan
 004                               column=Personal:pname, timestamp=1406299268535, value=amudhan
 004                               column=Personal:pname, timestamp=1406299225754, value=alex
 005                               column=Personal:pname, timestamp=1406299275355, value=alex
5 row(s) in 0.0320 seconds
```

Shell Commands - contd

- **Delete –** Deleting a single column
- Rowkey = 001 , CF = Personal , CQ = pname

Command: **delete 'Patient' , '001' , 'Personal:pname'**

```
hbase(main):004:0> delete 'Patient','001','Personal:pname'
0 row(s) in 0.0050 seconds

hbase(main):005:0> scan 'Patient'
ROW                                   COLUMN+CELL
 002                                  column=Personal:pname, timestamp=1406275812618, value=gowtham
 003                                  column=Personal:pname, timestamp=1406275824869, value=senthil
 004                                  column=Personal:age, timestamp=1406276092029, value=29
 004                                  column=Personal:pname, timestamp=1406275834026, value=siva
 005                                  column=Personal:pname, timestamp=1406275845665, value=amuthan
 006                                  column=Personal:pname, timestamp=1406275908785, value=alex
5 row(s) in 0.0390 seconds
```

DATA DOTZ

Shell Commands - contd

- **Delete –** Deleting a entire row
- Rowkey = 001 , CF = Personal , CQ = pname

Command: **deleteall 'Patient' , '004'**

```
hbase(main):027:0> scan 'Patient'
ROW                                   COLUMN+CELL
 002                                  column=Personal:pname, timestamp=1406275812618, value=gowtham
 003                                  column=Personal:pname, timestamp=1406275824869, value=senthil
 004                                  column=Personal:age, timestamp=1406276092029, value=29
 004                                  column=Personal:pname, timestamp=1406275834026, value=siva
 005                                  column=Personal:pname, timestamp=1406275845665, value=amuthan
 006                                  column=Personal:pname, timestamp=1406275908785, value=alex
5 row(s) in 0.0160 seconds

hbase(main):028:0> deleteall 'Patient','004'
0 row(s) in 0.0050 seconds

hbase(main):029:0> scan 'Patient'
ROW                                   COLUMN+CELL
 002                                  column=Personal:pname, timestamp=1406275812618, value=gowtham
 003                                  column=Personal:pname, timestamp=1406275824869, value=senthil
 005                                  column=Personal:pname, timestamp=1406275845665, value=amuthan
 006                                  column=Personal:pname, timestamp=1406275908785, value=alex
4 row(s) in 0.0120 seconds
```

Shell Commands - contd

- **Disable –** disable table
- Command: **disable 'Patient'**
- **Enable –** disable table
- Command: **enable 'Patient'**

```
hbase(main):030:0> disable 'Patient'
0 row(s) in 1.1900 seconds

hbase(main):031:0> enable 'Patient'
0 row(s) in 1.2470 seconds
```

www.datadotz.com

Shell Commands - contd

- **Alter –** Alter the table
- Command: **alter 'Patient',{NAME => 'Personal',VERSIONS => 5}**

```
hbase(main):041:0> disable 'Patient'
0 row(s) in 1.3660 seconds

hbase(main):042:0> alter 'Patient',{NAME => 'Personal', VERSIONS => 5}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 1.1800 seconds

hbase(main):043:0> enable 'Patient'
0 row(s) in 1.2110 seconds
```

Shell Commands - contd

- **Drop –** drop the table
- Command: **drop 'Patient'**

```
hbase(main):048:0> drop 'Patient'

ERROR: Table Patient is enabled. Disable it first.'

Here is some help for this command:
Drop the named table. Table must first be disabled: e.g. "hbase> drop 't1'"


hbase(main):049:0> disable 'Patient'
0 row(s) in 1.2110 seconds

hbase(main):050:0> drop 'Patient'
0 row(s) in 1.0970 seconds
```

Hbase Daemons

- HMaster
- HRegionServer
- HQuorumPeer

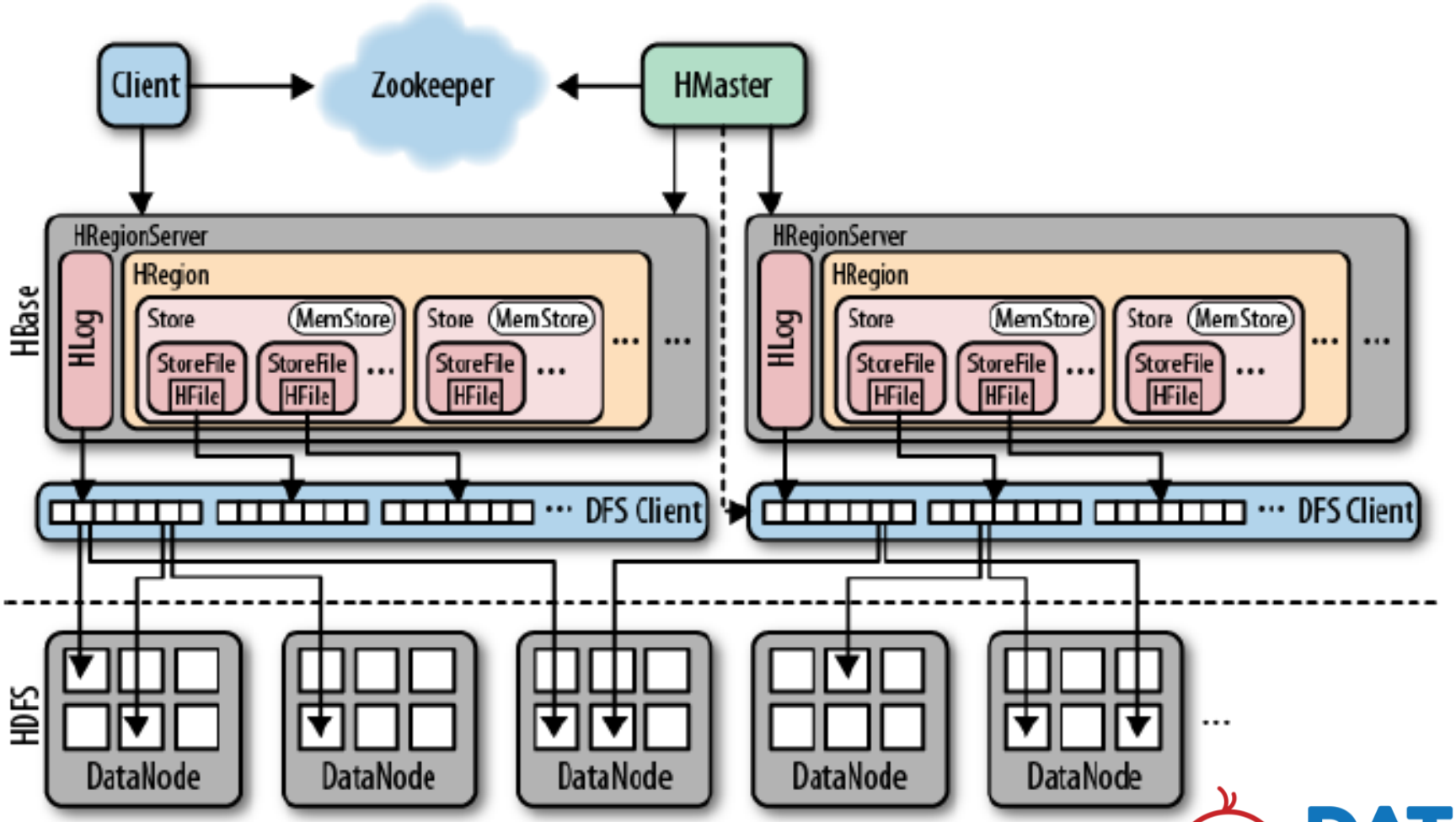
www.datadotz.com

Terms

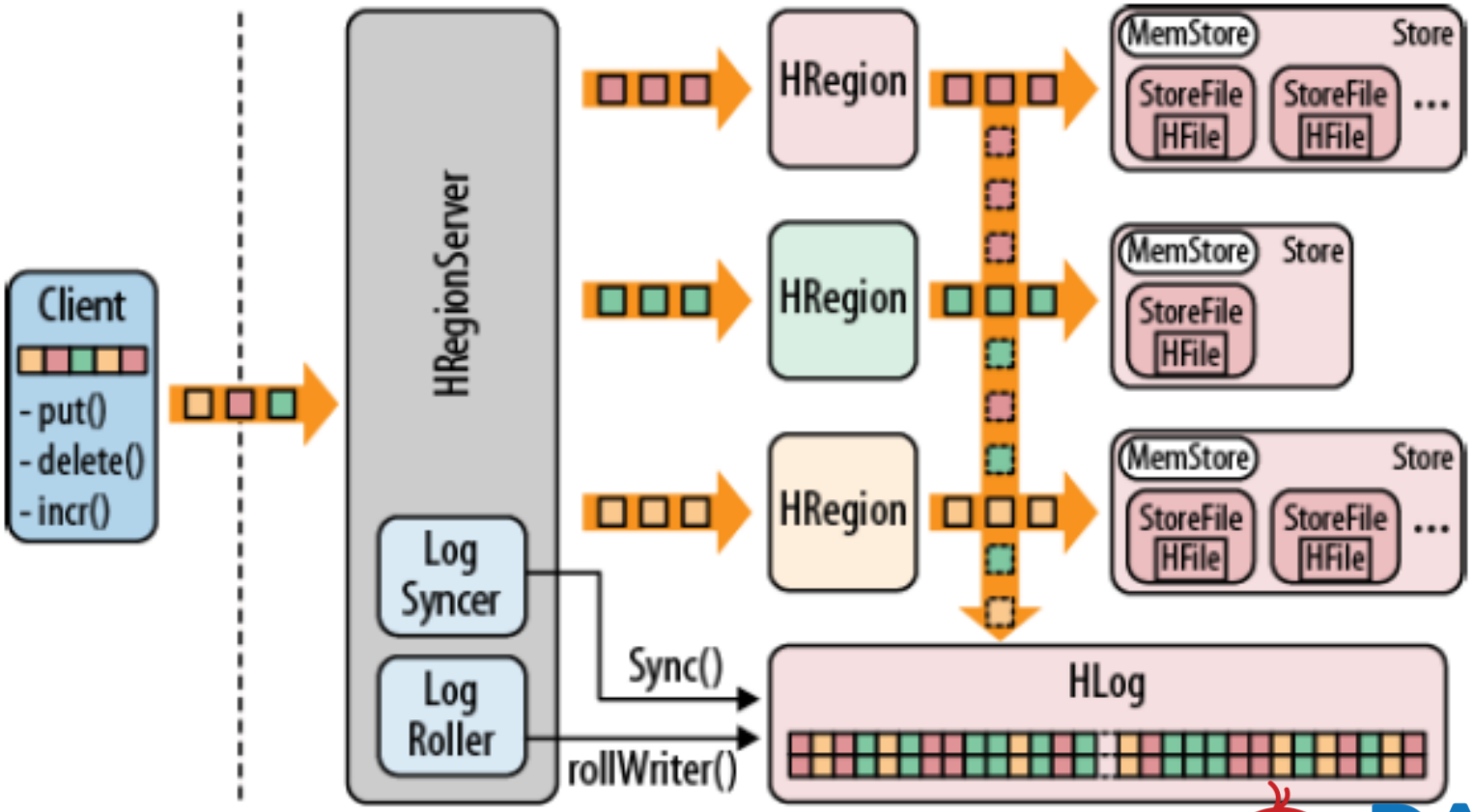
- Region
  - A subset of a table's rows , like a partition
- RegionServer
  - Serves data for reads and writes
- Master
  - Responsible for coordinating the slaves (HRegionServer)
  - Assigns regions, detects failures of HregionServers and controls some admin functions

www.datadotz.com

Hbase High Level Architecture



Write Ahead Log Flow



Finding a row

- Zookeeper
  - Stores global information about the cluster
- -ROOT-
  - A table that lists the .META. tables
- .META.
  - A table that lists all the regions and their locations

www.datadotz.com

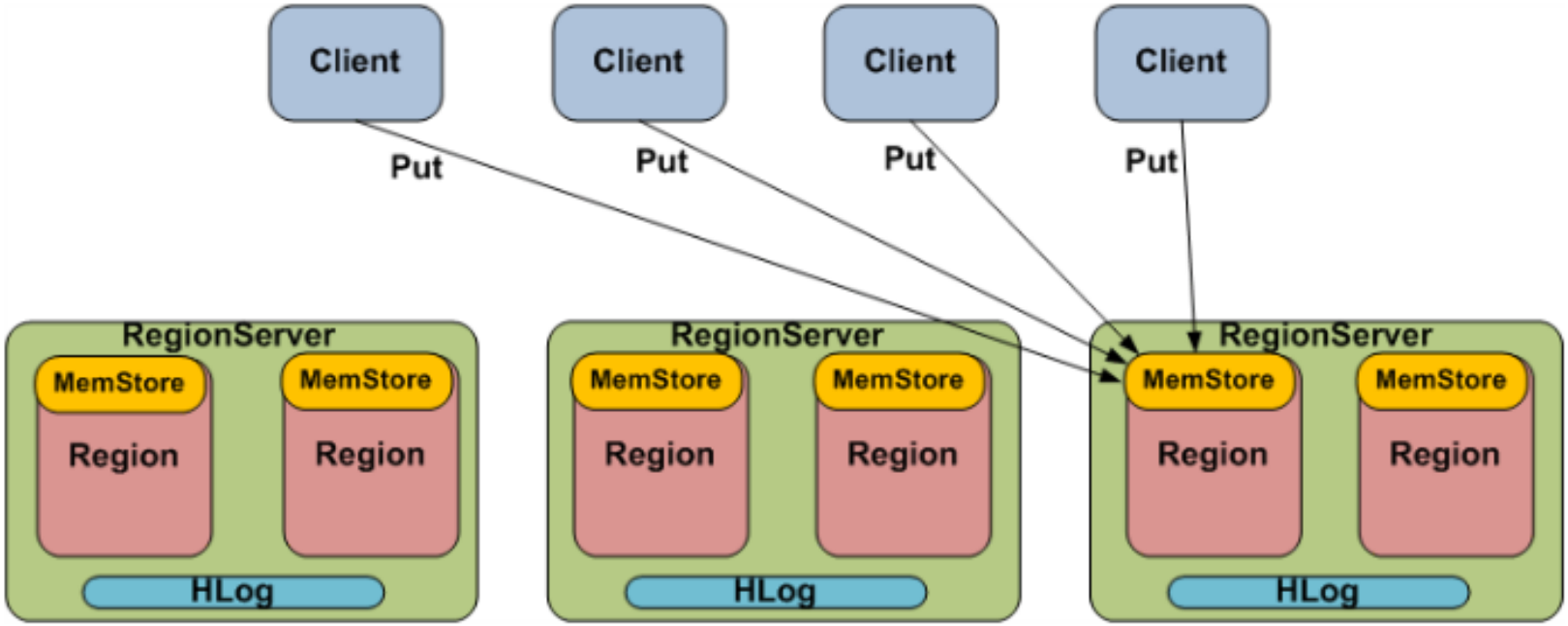
Caching Metadata

- Client caches information from zookeeper
  - Location of META and previously used Regions
- The cache can become stale
  - Regions may have moved
  - RegionServers can be unavailable

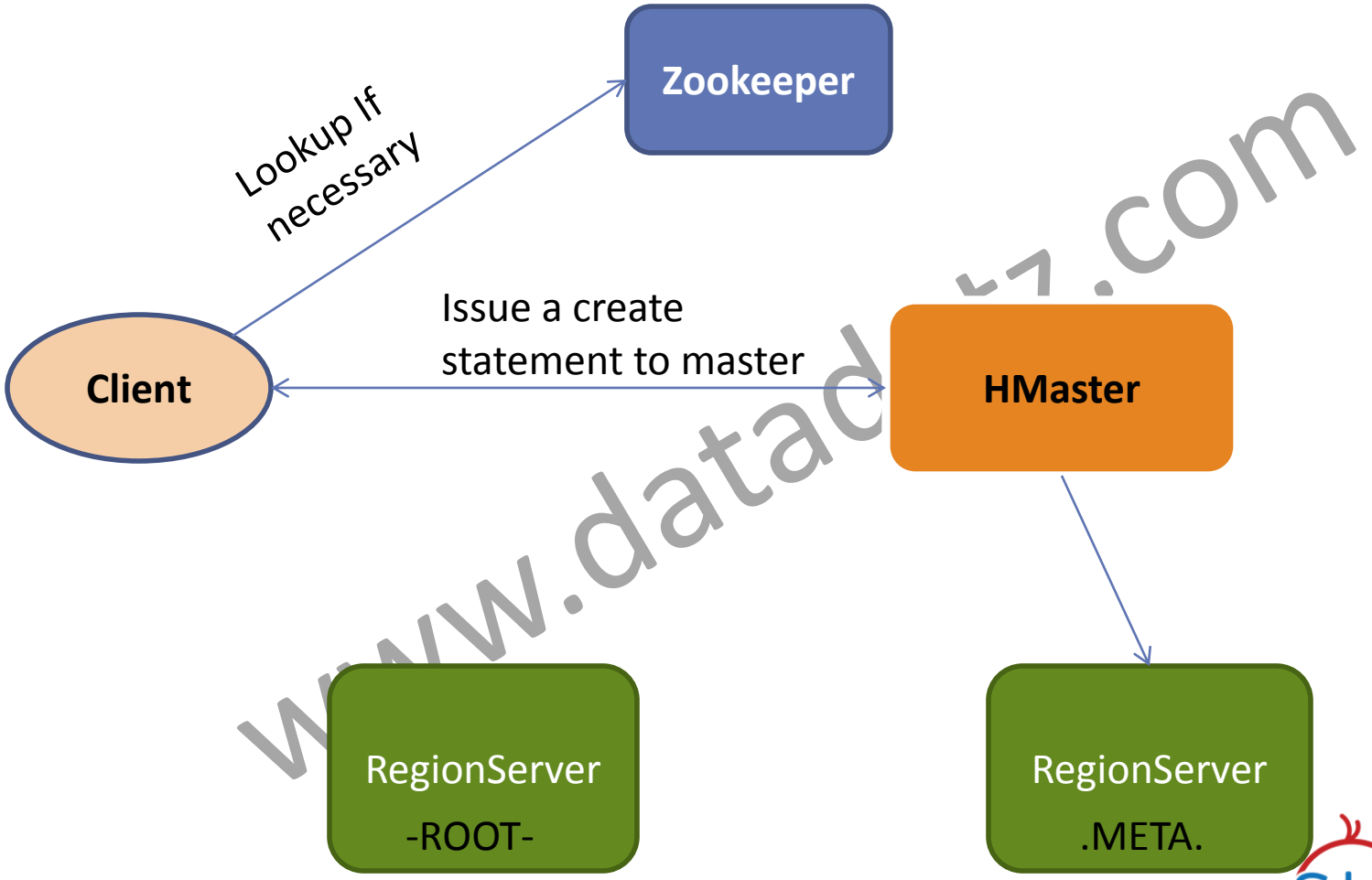
www.datadotz.com

Distribution of DATA

- Sequential RowKeys will go the same region



Create a table in HBase



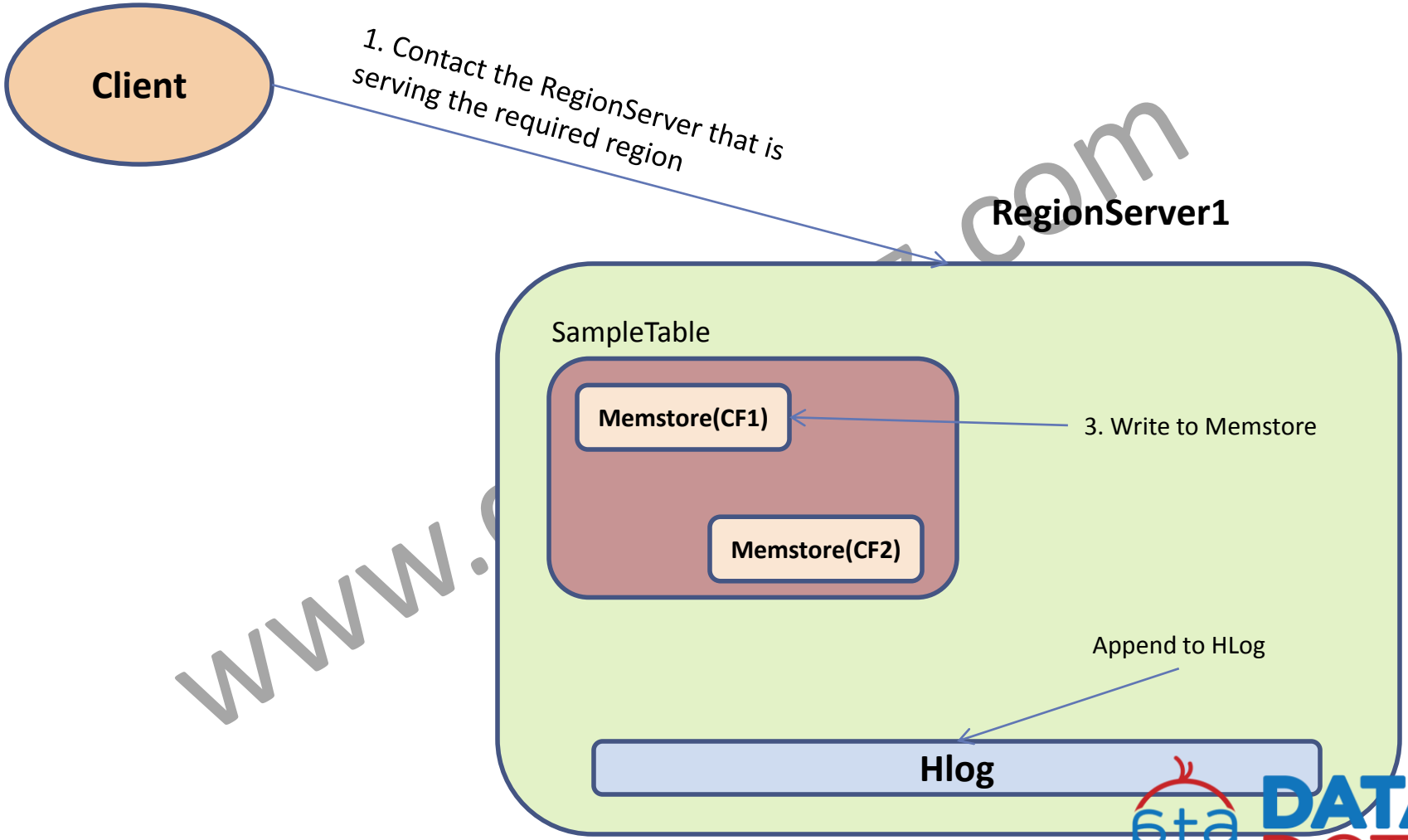
www.datadotz.com

Data Storage

- Data is written to the Region's Memstore
  - The Hlog is required for crash recovery if the MeMStore is lost
- When the Memstore gets to a certain size, it will flush to an immutable file (Store file)
- Eventually these store files will be aggreagted and cleaned up during a major compaction

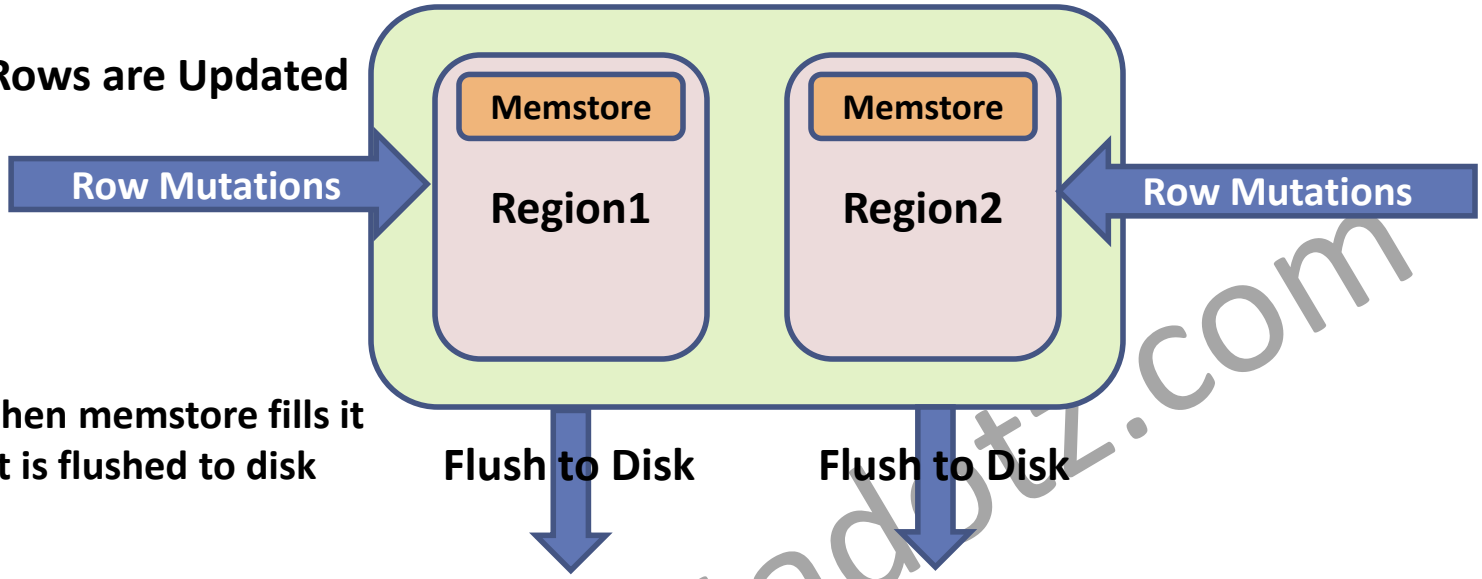
www.datadotz.com

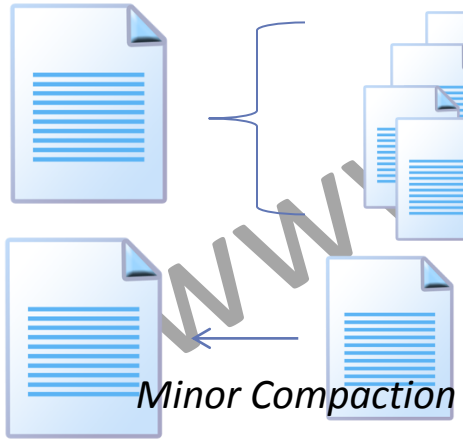
Modifying a Row in the Table



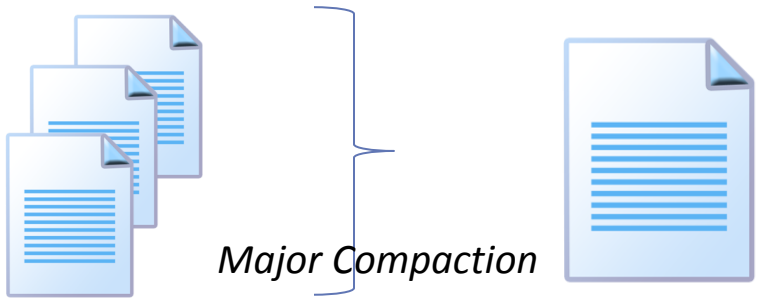
www.datadotz.com

Major and Minor Compaction





3. Several small store files are compacted together



4. Each Day the store files are compacted into a single file and cleaned up

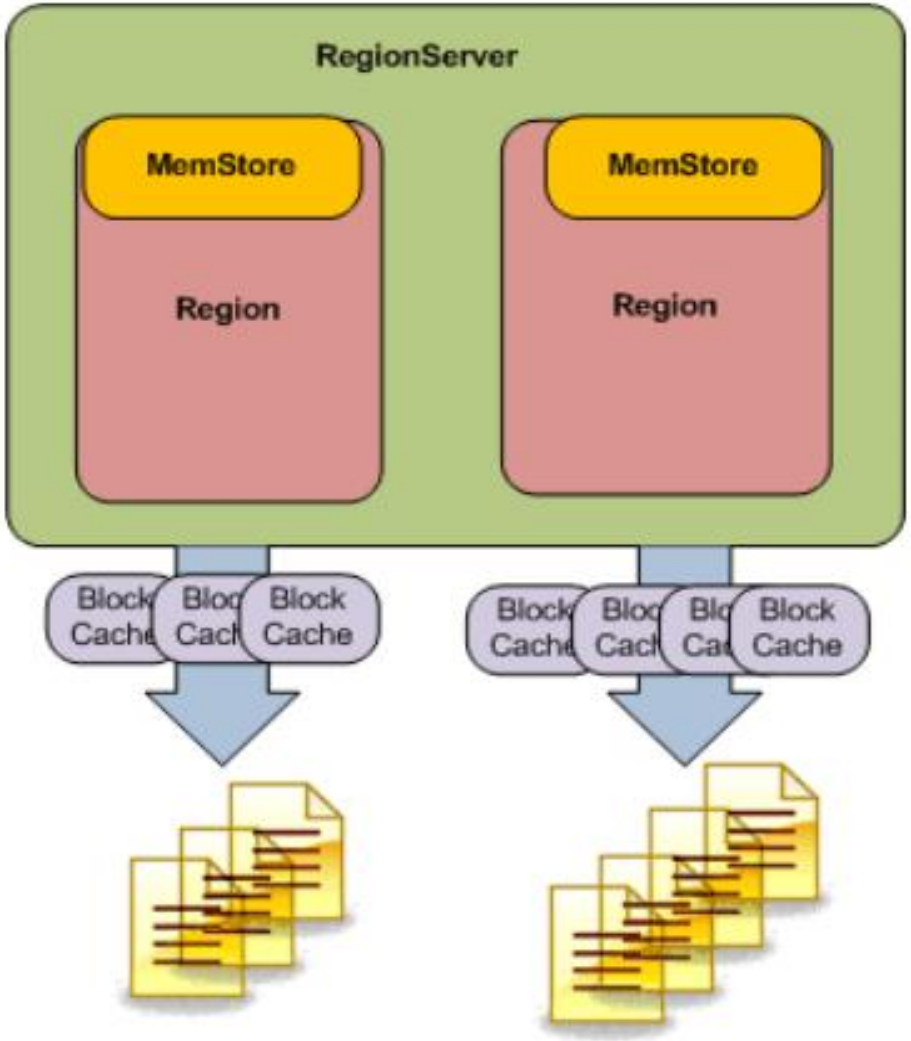
www.datadotz.com

HDFS locality

- The RegionServers might not be colocated with the data
- On Hbase startup, Regions are randomly assigned
  - Not ideal because most data blocks will not be local
- Compactions help
  - New Store files are written out to HDFS
  - HDFS will write the data locally

www.datadotz.com

Block Cache



DATA DOTZ

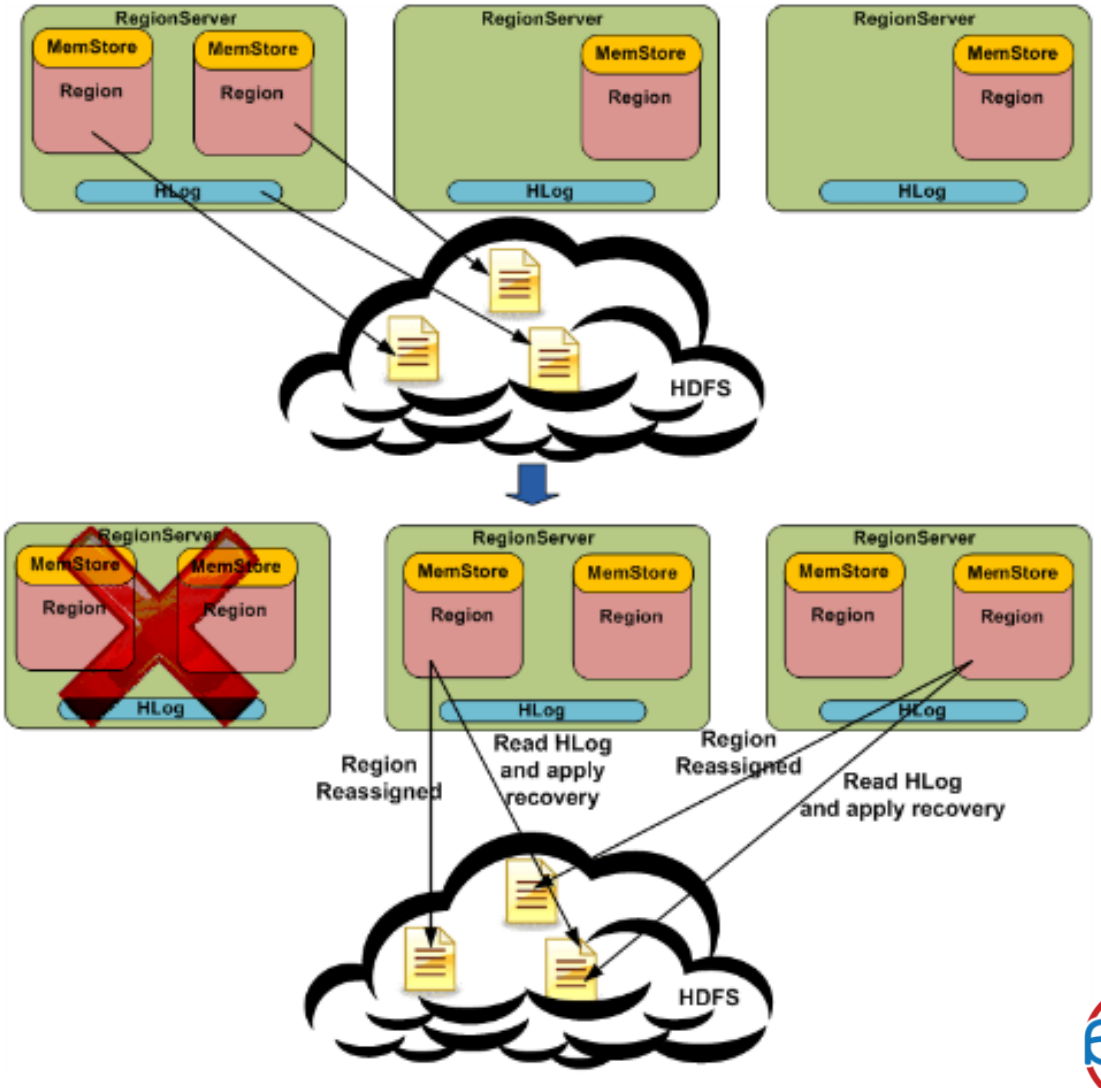
Region Splits

- When a Region grows to a certain size, the RegionServer will split the Region:
  - A Region reaches 256MB
  - RegionServer splits the Region into 2 roughly equal Regions
  - RegionServer updates META table
  - RegionServer informs Master of the split
  - Master assigns the new Region to a RegionServer

  Region Size is configurable.

www.datadotz.com

Crash Recovery



Hbase Java API

Configuration

HBaseConfiguration

HBaseAdmin

Htable

Put

Get

Scan

Result

ResultScanner

Filter

FilterList

www.datadotz.com

DATA DOTZ

HBaseAdmin

- Create Tables
- List Tables
- Delete Tables
- Check Status of Cluster
- Shut down the cluster

www.datadotz.com

DATA DOTZ

Administrative classes

- HBaseAdmin – main adminstrative class
- HTableDescriptive – contains name of table and its column families
- HColumnDesciptor – defines the attributes for a column family

www.datadotz.com

DATA DOTZ

Using HBaseAdmin

```
HbaseAdmin admin = new HBaseAdmin(conf);
HTableDescriptor tab = new HTableDescriptor(“newTable”);
HColumnDescriptor family = new HColumnDescriptor(“colfam”);
T.addFamily(family);
Admin.createTable(t);
```

www.datadotz.com

HTable

```
// open the table

Htable table = new Htable(conf, “sampleTable”);
```

www.datadotz.com

DATA DOTZ

CRUD

- Create – Put
- Read – Get, Scan
- Update – Put
- Delete - Delete, DeleteAll, Truncate

www.datadotz.com

DATA DOTZ

Put

```
Put p = new Put(Bytes.toBytes("row_key"));

p.add(Bytes.toBytes("cf1")),Bytes.toBYtes("cq1").
Bytes.toBYtes("val"));

table.put(p);
```

www.datadotz.com

Get

```
Get g = new Get(Bytes.toBytes("row_key"));

Result row = table.get(g);

Byte[] value =
row.getValue(Bytes.toBytes("cf1"),Bytes.toBytes("cq1"));
```

www.datadotz.com

Result

```
NavigableMap<byte[], byte[]> map =
row.getFamilyMap(Bytes.toBytes("cf1"));

For(byte[] col: map.keyset()){
        byte[] val = map.get(Bytes.toBytes('col'));
        String valueStr = Bytes.toString(val);
        String keyStr = Bytes.toString(row.getRow());

}
```

www.datadotz.com

Scan

```
scan s = new scan();
ResultScanner scanner = table.getScanner(s);

for (Result r : scanner){
// process the result
}
```

www.datadotz.com

DATA DOTZ

Other Scan methods

- addFamily(Bytes.toBytes(“cf1”));
- addColumn(Bytes.toBytes(“cf1”), Bytes.toBytes(“cq1”));
- setMaxVersions(3);
- setTimeRange(minTimestamp,maxTimestamp);
- setStartRow(rowkey);
- setStopRow(rowkey);

www.datadotz.com

Getting Sorted Values

```
for ( Result rr : scanner){
        for (KeyValue kv : rr.sorted()){
                byte[] value = kv.getValue();
                String key =
        Bytes.toStringBinary(kv.getRow());
                String v = Bytes.toString(value);
}
```

www.datadotz.com

DATA DOTZ

Other useful HTable functions

- setAutoFlush(boolean);
- flushCommits();
- setWriteBufferSize(long);

www.datadotz.com

DATA DOTZ

Filter

- Used to restrict the rows from a scanner
- RowFilter
  - Filter by row key
- ValueFilter
  - Filter by cell value
- PageFilter
  - Limits the number of rows returned

www.datadotz.com

Disable the WAL

- Disable the write ahead log for faster writes
  - *Put.setWriteToWAL(boolean)*
  - If a server crashes, edits will be lost

www.datadotz.com

DATA DOTZ

REST

- Stargate can run as a daemon which starts an embedded Jetty servlet container and deploys the servlet into it.
  - ./bin/hbase-daemon.sh start rest -p <port>
  - curl http://localhost:8000/version
  - curl http://localhost:8000/version/cluster
  - curl http://localhost:8000/status/cluster
  - curl http://localhost:8000/
  - curl http://localhost:8000/content/schema

www.datadotz.com

DATA DOTZ

Thrift

- Expose services (such as HBase) to applications written in other languages such as pyhon, perl, c++ and ZzRuby
- Start the daemon
  - Bin/hbase-daemon.sh start thrift
- Generate the language-specific classes
  - Thrift –gen py Hbase.thrift
- Look at Hbase.thrift for the set of methods available such as getTableNames, createTable, getRow, mutateRow

www.datadotz.com

DATA DOTZ

Hbase and MapReduce

- Hbase tables as input or output of a Mapreduce job
- TableInputFormat
  - Splits Hbase tables on their Regions
  - Uses a scanner to read each split
- TableOutputFormat
  - Uses Put to write records
- HFileOutputFormat
  - Writes data in the native Hfile format (for bulk load)

www.datadotz.com

Tuning FileSystem Properties

- dfs.datanode.max.xcievers
  - Default – 256; increase to 2048
  - Max no of files that HDFS can server concurrently
- ulimit -n

www.datadotz.com

DATA DOTZ

Performance Considerations

- Scanner Caching
  - Hbase.client.scanner.caching
  - setScannerCaching()
- Compression of store files
  - Reduces the bandwidth and disk space needed while not incurring much penalty
  - Hbase ships with Gzip, but LZO is better
- Hfile Block size
  - Hfile.min.blocksize.size

www.datadotz.com

DATA DOTZ

Allocating RAM

- Allocate up to 8-12 GB of heap per RegionServer
  - By default, block cache uses 20% and memstore uses 40%
  - For read heavy workloads, consider a smaller memstore in lieu of more block cache
  - For machines with >12GB of Ram, the filesystem cache may be useful

www.datadotz.com

Who is using HBase

- http://wiki.apache.org/hadoop/Hbase/PoweredBy

www.datadotz.com

DATA DOTZ

Thank You !!!

www.datadotz.com