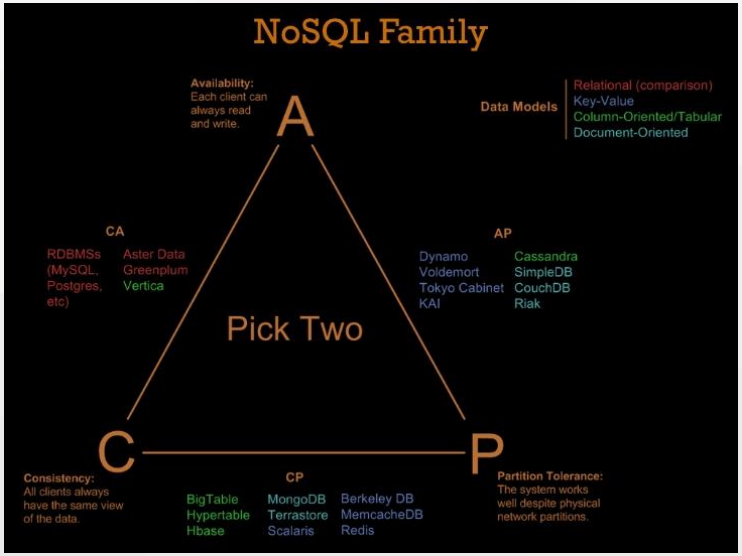
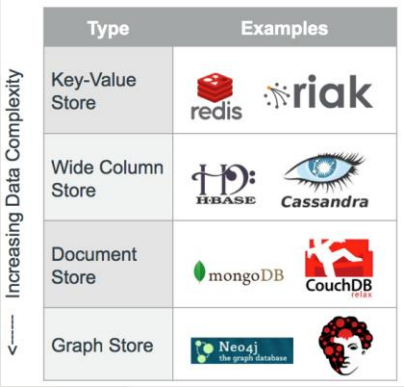



# The Pilot Of HBase

2017.5 XenRon

# CONTENTS






Review

# Review



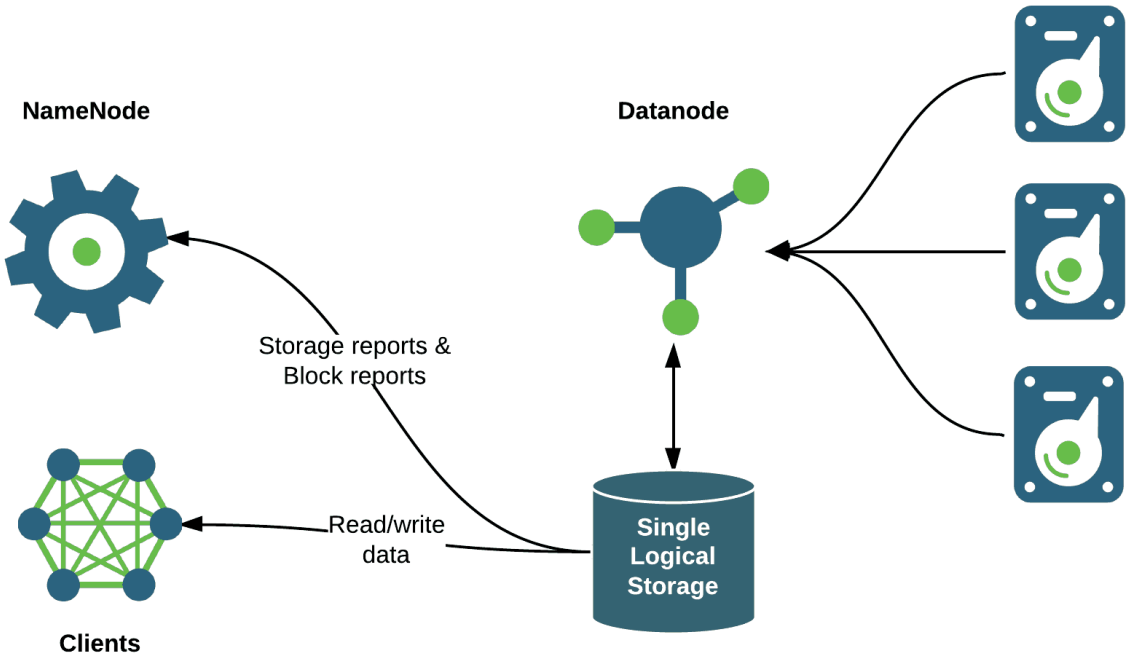



**Figure 1: A DataNode presented itself as a single logical storage**

# HDFS



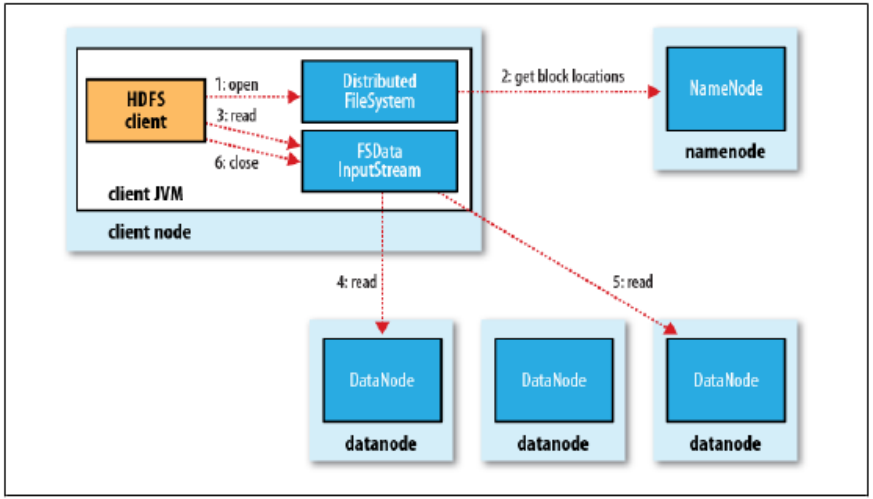



*Figure 3-2. A client reading data from HDFS*

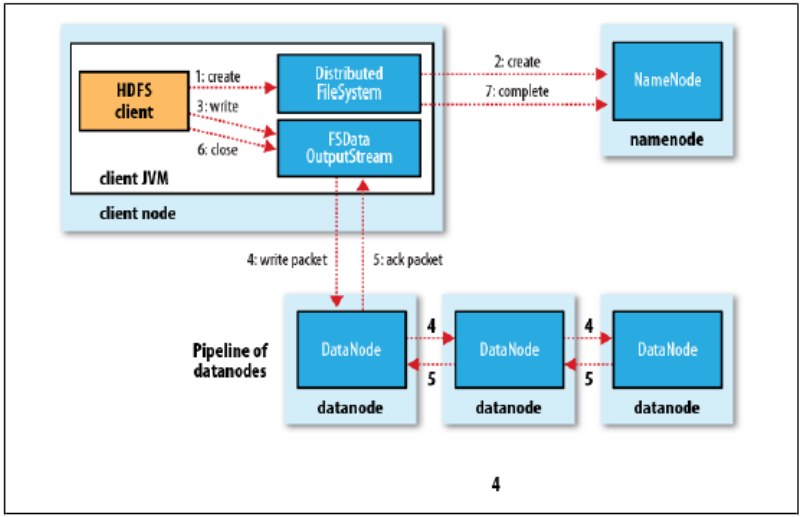
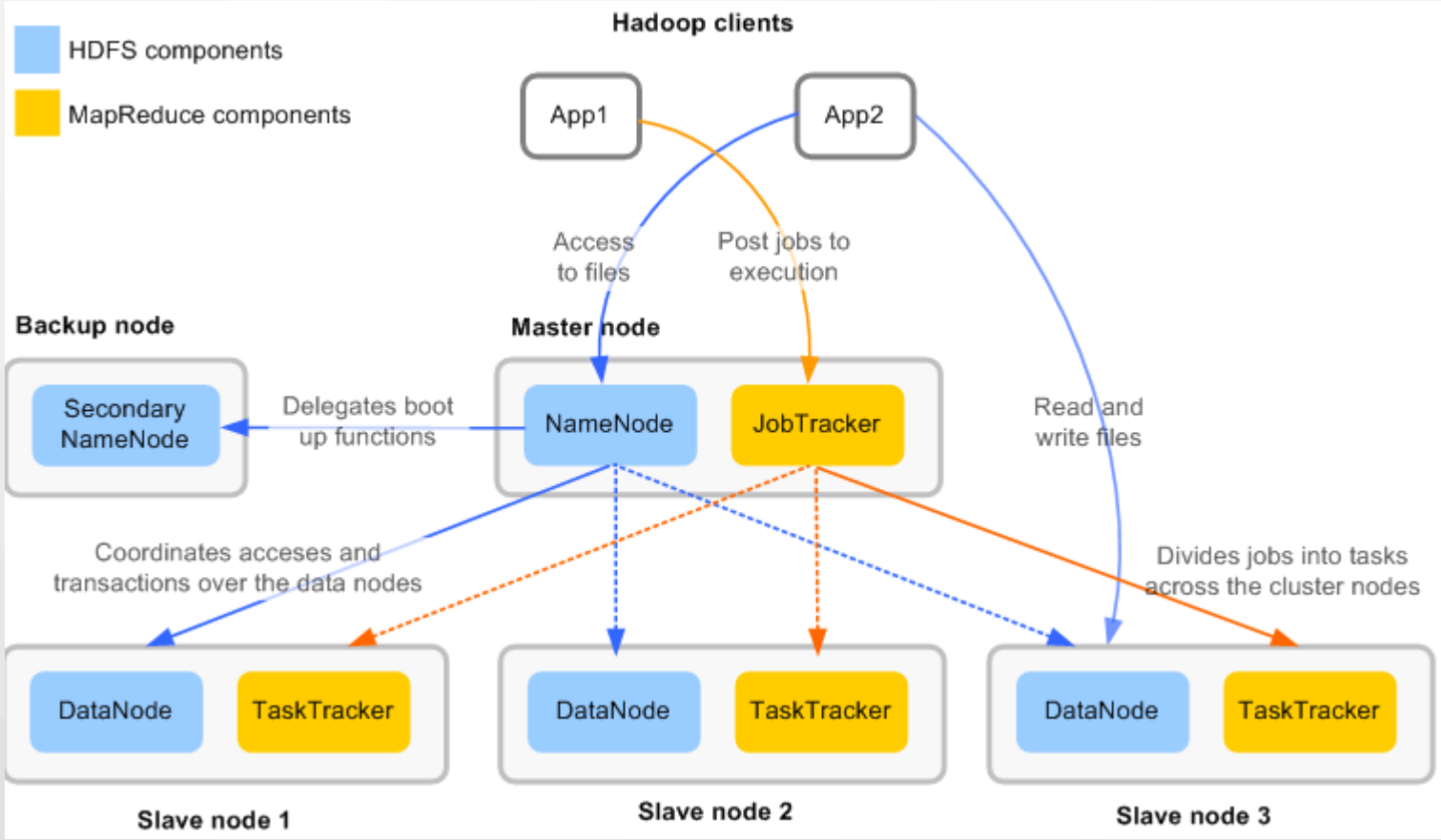



*Figure 3-4. A client writing data to HDFS*

# Map Reduce

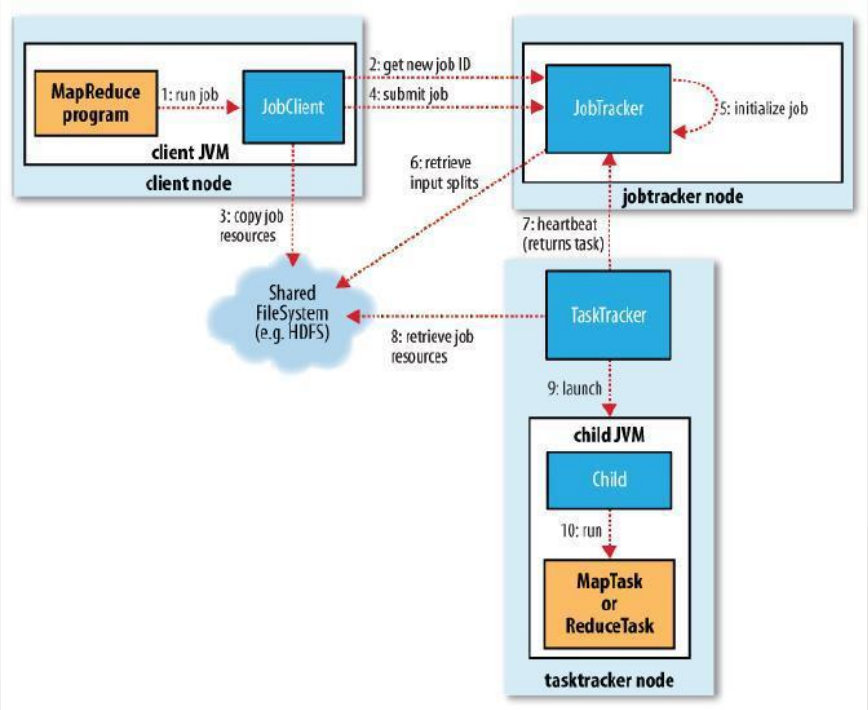



HDFS components
MapReduce components
Hadoop clients
App1
App2
Access to files
Post jobs to execution
Backup node
Master node
Secondary NameNode
Delegates boot up functions
NameNode
JobTracker
Read and write files
Coordinates acceses and transactions over the data nodes
Divides jobs into tasks across the cluster nodes
DataNode
TaskTracker
DataNode
TaskTracker
DataNode
TaskTracker
Slave node 1
Slave node 2
Slave node 3

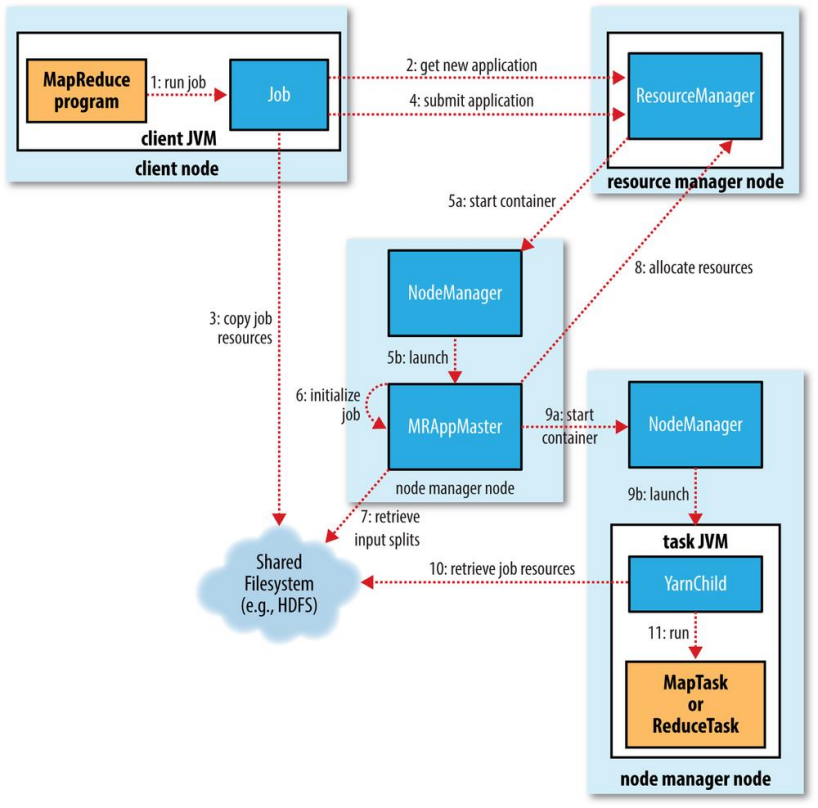
# Map Reduce



Hadoop 1.x




Hadoop 2.x



# Map Reduce Model #1
# Iterator Map-Reduce



A Project by Brad Lyon
Want an App for this? Tell me

## Exploration of the Google PageRank Algorithm

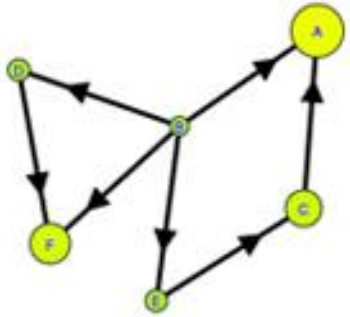
Circles correspond to web pages, links correspond to hyperlinks

Questions/Comments
A Few Notes About This

Damping Factor d:
0.85

Generate New

- *Drag nodes to rearrange*
- *Click on node, then click on another to create link*
- *Double-click to add node (max 14)*
- ***Del+mouseover** deletes node*
- *Click on link to delete*
- *Mouseover link to see location in matrix*




**Sorted PageRank Vector**
(8 Iterations)

Explore Convergence

Click here to explore/hide how the sequence converged to the solution

| | |
|---|---|
| C | 0.19 |
| D | 0.11 |
| E | 0.11 |
| B | 0.09 |

Hide Matrix Stuff

The PageRank vector **x** satisfies **x=Gx**, where

Google Matrix **G** = d * [ (Hyperlink Matrix **H**) + (Dangling Nodes Matrix **A**) ] + ((1-d)/N) * (NxN Matrix **U** of all 1's)

$$G = 0.85 * \left[ H + A \right] + \frac{0.15}{6} * U$$

**Hyperlink Matrix H**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | 1/4 | 1 | | | |
| B | | | | | | |
| C | | | | | 1 | |
| D | | 1/4 | | | | |
| E | | 1/4 | | | | |
| F | | 1/4 | | 1 | | |

**Dangling Nodes Matrix A**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | 1/6 | | | | 1/6 |
| B | | 1/6 | | | | 1/6 |
| C | | 1/6 | | | | 1/6 |
| D | | 1/6 | | | | 1/6 |
| E | | 1/6 | | | | 1/6 |
| F | | 1/6 | | | | 1/6 |

**Matrix U with all 1's**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 1 | 1 | 1 |
| B | 1 | 1 | 1 | 1 | 1 | 1 |
| C | 1 | 1 | 1 | 1 | 1 | 1 |
| D | 1 | 1 | 1 | 1 | 1 | 1 |
| E | 1 | 1 | 1 | 1 | 1 | 1 |
| F | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 0.17 | 0.24 | 0.88 | 0.03 | 0.03 | 0.17 |

# Map Reduce Model #2
# Job Control



## Naïve Bayes in MapReduce

- Map
  - Input data $\{x, y\}$ from a subgroup of data
  - Output: 3 types of keys

$$key = (x_j = a_{pj}^j, y = c_k), value = \sum_{subgroup} 1(x_j = a_{pj}^j \mid y = c_k)$$

$$key = (y = c_k), value = \sum_{subgroup} 1(y = c_k)$$

$$key = "samples", value = \sum_{subgroup} 1$$

- Reduce
  - Sum all the values of each key
  - Compute the conditional and marginal probabilities

## Support Vector Machine in MapReduce

- Map
  - Input: $\{(x, y)\}$
  - Output:

$$key = GGW, value = 2w + 2C \sum_{subgroup} (w.x_i - y_i) x_i$$

- Reduce
  - Aggregate the values of gradient from all mappers
  - Update

$$w = w - \eta * \nabla G_w$$

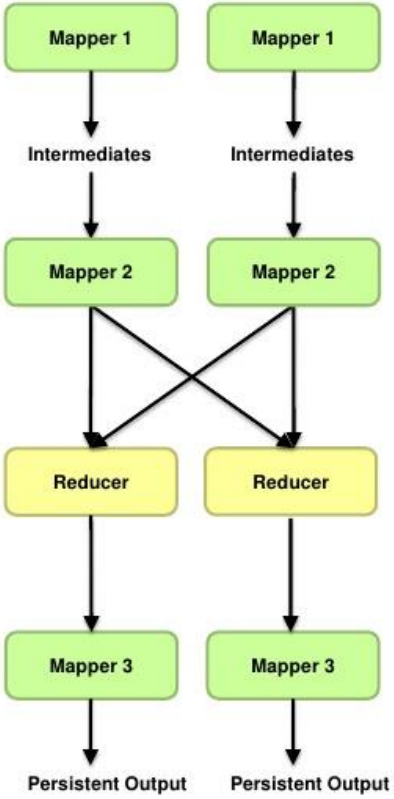
- Driver program that sets up the iterations and checks for convergence

# Map Reduce Model #3
# Chain-Mapper Chain-Reduce



## Chaining in Hadoop

- Map+ Reduce Map*
  - 1 or more Mappers
    - Can use IdentityMapper
  - 1 reducer
    - No reducers: conf.setNumReduceTasks(0)?
  - 0 or more Mappers
- Usual *combiners* and *partitioners*
- By default, data passed between Mappers by usual writing of intermediate data to disk
  - Can always use side-effects…
  - There is a better, built-in way to bypass this and pass (Key,Value) pairs *by reference* instead
    - Requires different Mapper semantics!



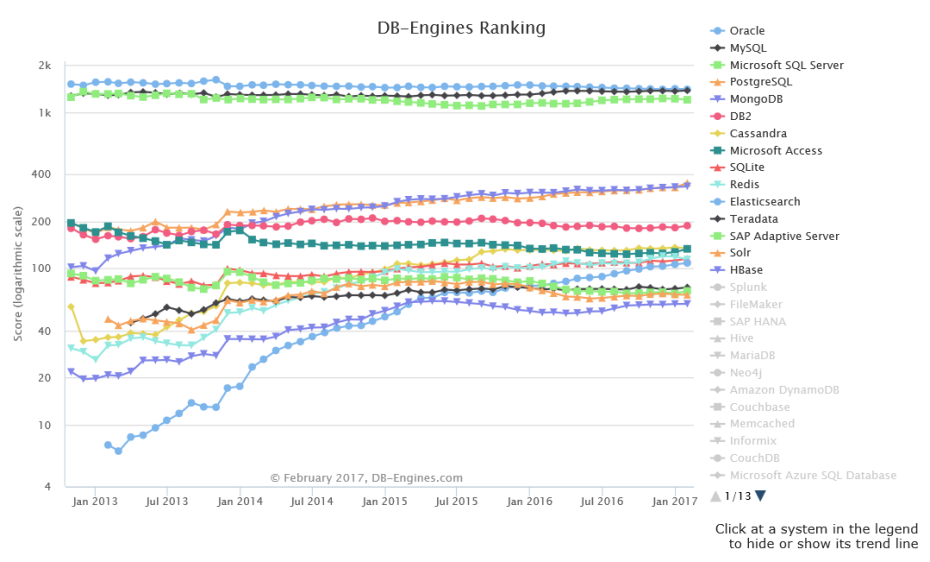

# NoSQL

# DB-Engines Ranking



318 systems in ranking, February 2017

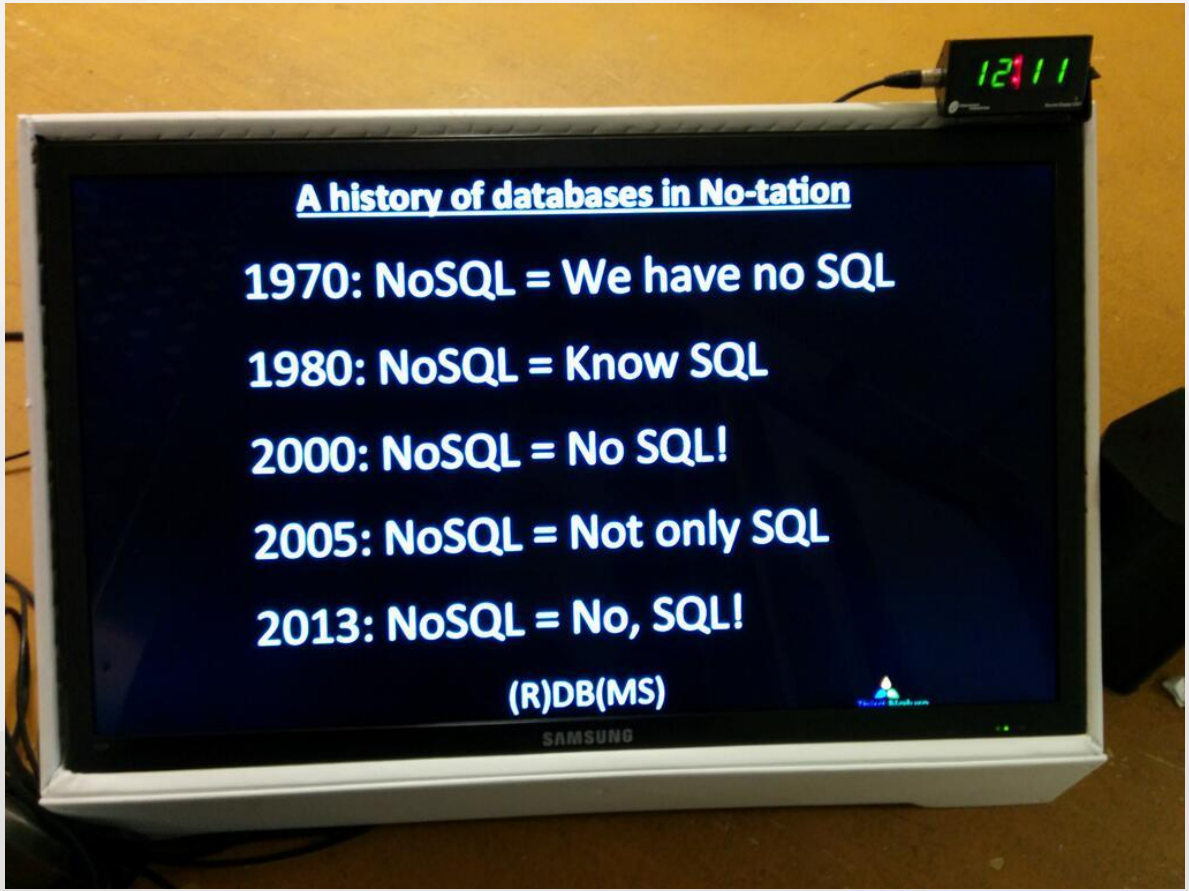
| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Feb 2017 | Jan 2017 | Feb 2016 | | | Feb 2017 | Jan 2017 | Feb 2016 |
| 1. | 1. | 1. | Oracle | Relational DBMS | 1403.83 | -12.89 | -72.31 |
| 2. | 2. | 2. | MySQL | Relational DBMS | 1380.30 | +14.02 | +59.18 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational DBMS | 1203.45 | -17.50 | +53.23 |
| 4. | ↑ 5. | ↑ 5. | PostgreSQL | Relational DBMS | 353.68 | +23.31 | +65.02 |
| 5. | ↓ 4. | ↓ 4. | MongoDB | Document store | 335.50 | +3.60 | +29.90 |
| 6. | 6. | 6. | DB2 | Relational DBMS | 187.90 | +5.41 | -6.58 |
| 7. | 7. | ↑ 8. | Cassandra | Wide column store | 134.38 | -2.06 | +2.62 |
| 8. | 8. | ↓ 7. | Microsoft Access | Relational DBMS | 133.39 | +5.94 | +0.31 |
| 9. | ↑ 10. | 9. | SQLite | Relational DBMS | 115.31 | +2.93 | +8.53 |
| 10. | ↓ 9. | 10. | Redis | Key-value store | 114.03 | -4.66 | +11.96 |
| 11. | 11. | ↑ 12. | Elasticsearch | Search engine | 108.31 | +2.14 | +30.47 |
| 12. | 12. | ↑ 13. | Teradata | Relational DBMS | 75.60 | +1.43 | +2.22 |
| 13. | 13. | ↓ 11. | SAP Adaptive Server | Relational DBMS | 71.74 | +2.63 | -8.30 |
| 14. | 14. | 14. | Solr | Search engine | 67.69 | -0.39 | -4.59 |
| 15. | 15. | ↑ 16. | HBase | Wide column store | 59.24 | +0.10 | +7.22 |
| 16. | 16. | ↑ 18. | Splunk | Search engine | 56.03 | +0.54 | +13.20 |
| 17. | 17. | 17. | FileMaker | Relational DBMS | 55.19 | +1.71 | +8.16 |
| 18. | 18. | ↑ 19. | SAP HANA | Relational DBMS | 52.45 | +0.52 | +14.37 |
| 19. | 19. | ↓ 15. | Hive | Relational DBMS | 47.95 | -3.19 | -4.83 |
| 20. | 20. | ↑ 23. | MariaDB | Relational DBMS | 45.35 | +0.31 | +16.57 |
| 21. | 21. | 21. | Neo4j | Graph DBMS | 36.27 | +0.00 | +3.98 |
| 22. | 22. | ↑ 26. | Amazon DynamoDB | Document store | 32.19 | +1.16 | +10.39 |
| 23. | 23. | ↑ 24. | Couchbase | Document store | 31.18 | +0.96 | +5.79 |
| 24. | 24. | ↓ 22. | Memcached | Key-value store | 30.53 | +2.09 | +1.60 |
| 25. | 25. | ↓ 20. | Informix | Relational DBMS | 27.25 | +0.82 | -5.76 |




http://db-engines.com/en/ranking
http://db-engines.com/en/ranking_trend

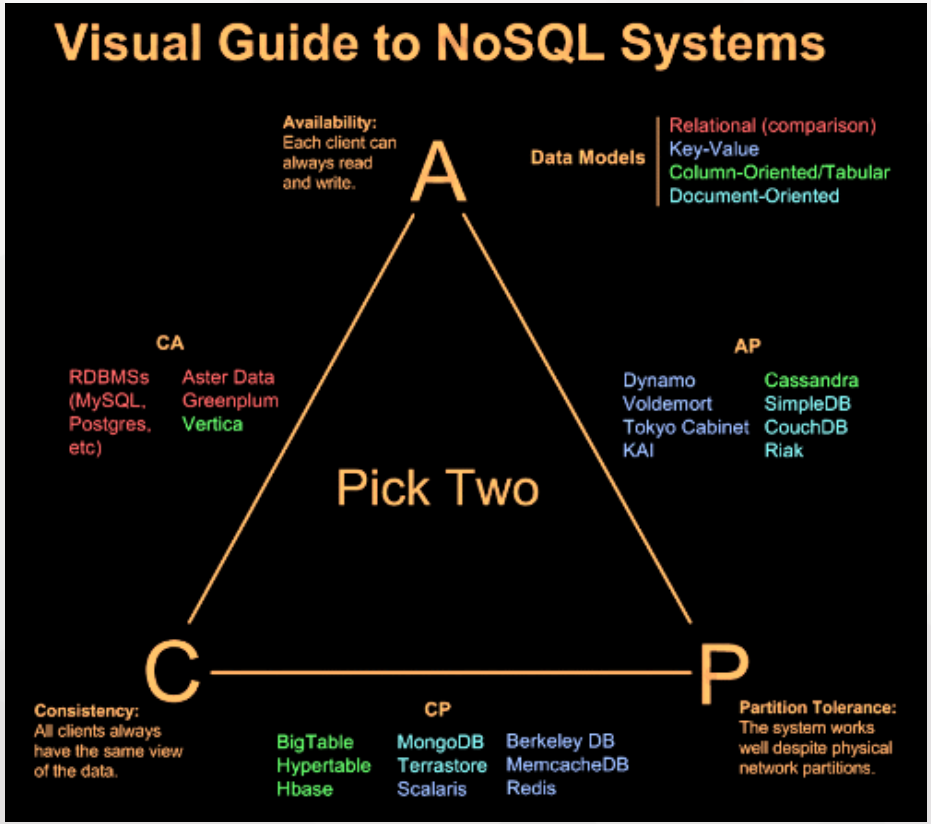
# The History of SQL





# CAP Theorem






Professor Eric A. Brewer
University of California at Berkeley

# Cassandra



## Cassandra Write Data Flows

### Single Region, Multiple Availability Zone

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk


If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

SSTable disk writes and compactions occur asynchronously

NETFLIX

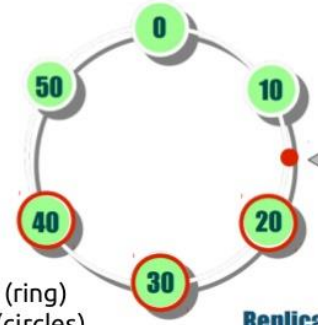
## Architecture

**Partitioner:**
Consistent Hashing

**Key:** "www.google.com"

md5("www.google.com")

14




**Replication Factor = 3**

- Cluster (ring)
- Nodes (circles)
- Peer-to-Peer Model
- Gossip Protocol

# MongoDB

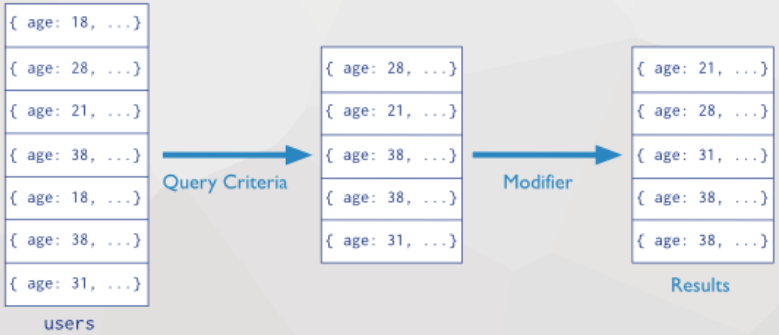


mongoDB Official Website : https://www.mongodb.org/
The latest stable Release : v3.0.4

```
{
  name: "sue",                    ← field: value
  age: 26,                        ← field: value
  status: "A",                    ← field: value
  groups: [ "news", "sports" ]    ← field: value
}
```

Collection — Query Criteria — Modifier

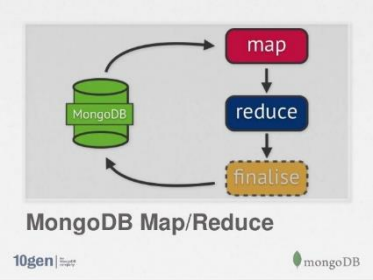
```
db.users.find( { age: { $gt: 18 } } ).sort( {age: 1 } )
```

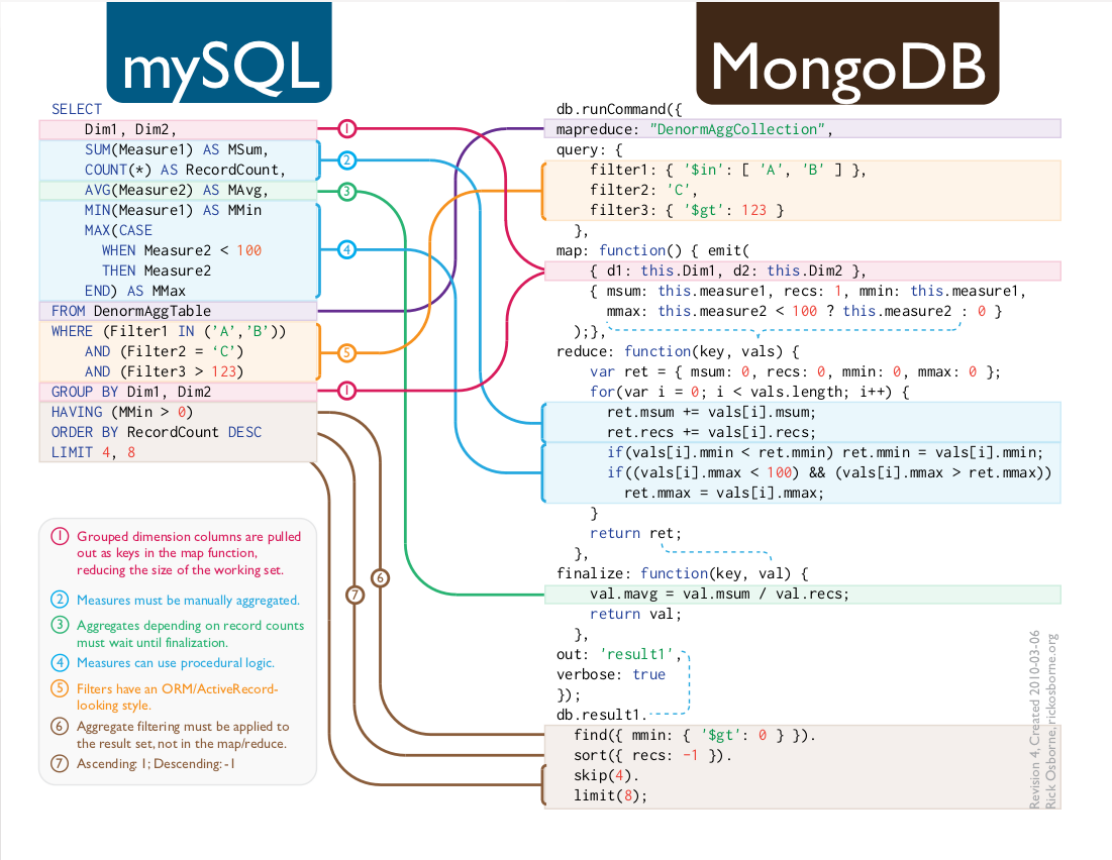



```
db.users.insert (                 ← collection
  {                               }
    name: "sue",                  ← field: value   } document
    age: 26,                      ← field: value   }
    status: "A"                   ← field: value   }
  }
)
db.users.update(                  ← collection
  { age: { $gt: 18 } },           ← update criteria
  { $set: { status: "A" } },      ← update action
  { multi: true }                 ← update option
)

db.users.remove(                  ← collection
  { status: "D" }                 ← remove criteria
)
```

# MongoDB




map
reduce
finalise
MongoDB
MongoDB Map/Reduce
10gen
mongoDB

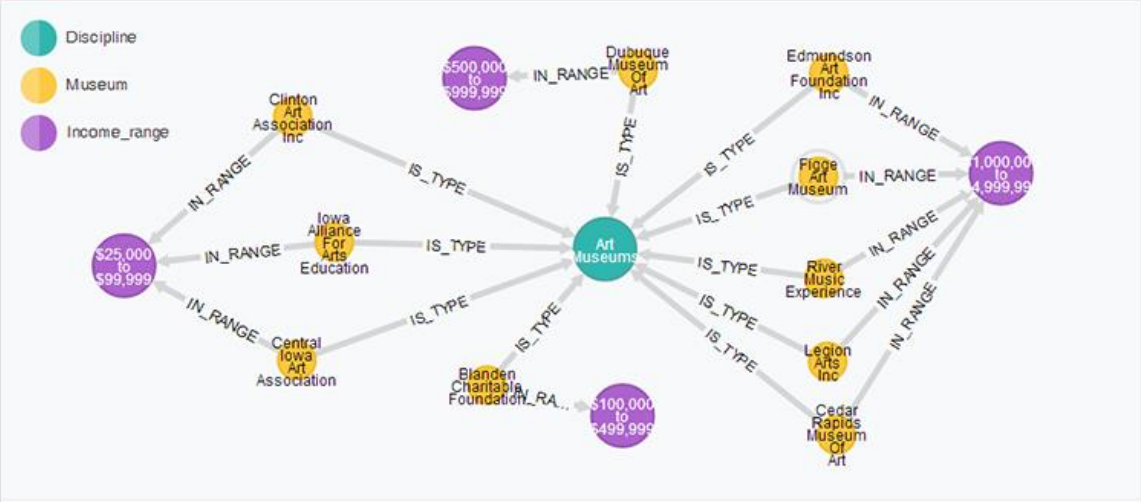

mySQL
MongoDB
SELECT
Dim1, Dim2,
SUM(Measure1) AS MSum,
COUNT(*) AS RecordCount,
AVG(Measure2) AS MAvg,
MIN(Measure1) AS MMin
MAX(CASE
WHEN Measure2 < 100
THEN Measure2
END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A','B'))
AND (Filter2 = 'C')
AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
db.runCommand({
mapreduce: "DenormAggCollection",
query: {
filter1: { '$in': [ 'A', 'B' ] },
filter2: 'C',
filter3: { '$gt': 123 }
},
map: function() { emit(
{ d1: this.Dim1, d2: this.Dim2 },
{ msum: this.measure1, recs: 1, mmin: this.measure1,
mmax: this.measure2 < 100 ? this.measure2 : 0 }
);},
reduce: function(key, vals) {
var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
for(var i = 0; i < vals.length; i++) {
ret.msum += vals[i].msum;
ret.recs += vals[i].recs;
if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
ret.mmax = vals[i].mmax;
}
return ret;
},
finalize: function(key, val) {
val.mavg = val.msum / val.recs;
return val;
},
out: 'result1',
verbose: true
});
db.result1.
find({ mmin: { '$gt': 0 } }).
sort({ recs: -1 }).
skip(4).
limit(8);
1 Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
2 Measures must be manually aggregated.
3 Aggregates depending on record counts must wait until finalization.
4 Measures can use procedural logic.
5 Filters have an ORM/ActiveRecord-looking style.
6 Aggregate filtering must be applied to the result set, not in the map/reduce.
7 Ascending: 1; Descending: -1
Revision 4, Created 2010-03-06
Rick Osborne, rickosborne.org

# Neo4j



```
// Iowa art museums with income ranges greater than $10,000
MATCH (inc_rng)<-[:IN_RANGE]-(lam:Museum {state: "IA"})-[:IS_TYPE]->
(disc:Discipline {code: "ART"})
WHERE toInt(inc_rng.code) > 1
RETURN lam, inc_rng, disc
```

CYPHER MATCH (inc_rng)<-[:IN_RANGE]-(lam:Museum {state: "IA"})-[:IS_TYPE]->(disc:Discipline {code: "ART"}) WHERE




Displaying 15 nodes, 20 relationships

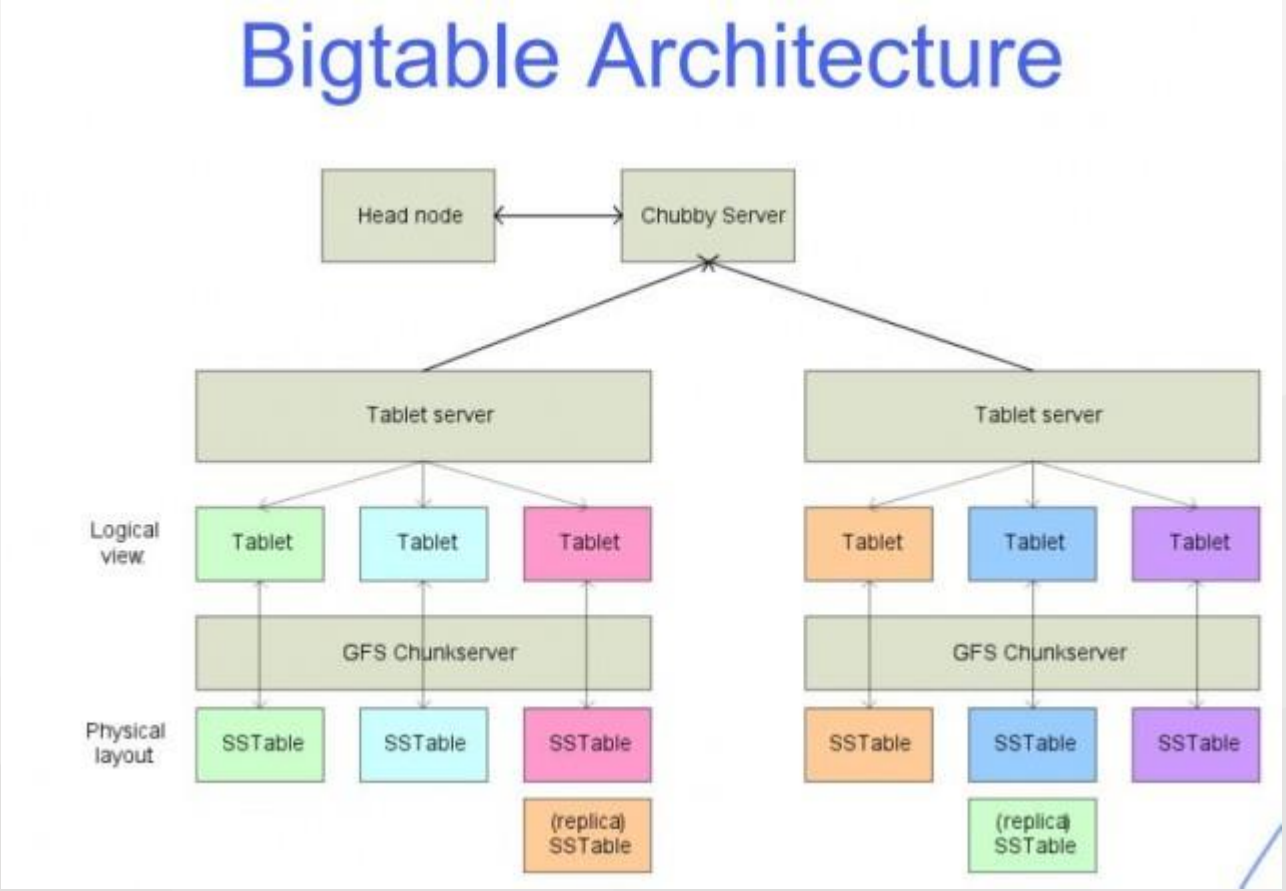
http://neo4j.com/docs/stable/cypher-query-lang.html

# PART2

## Hbase Architecture

# BigTable






https://research.google.com/archive/bigtable.html

# Java




**Java SE Public Updates**

| Major Release | GA Date | End of Public Updates Notification | End of Public Updates |
|---|---|---|---|
| 5.0 | May 2004 | Apr 2008 | Oct 2009 |
| 6 | Dec 2006 | Feb 2011 | Feb 2013 |
| 7 | Jul 2011 | Mar 2014 | Apr 2015 |
| 8 | Mar 2014 | TBD | Sep 2017* |

* or later, depending on factors described above.

**http://www.oracle.com/technetwork/java/eol-135779.html**

# HBase Versions



*Hadoop version support matrix*

- "S" = supported
- "X" = not supported
- "NT" = Not tested

| | HBase-0.94.x | HBase-0.98.x (Support for Hadoop 1.1+ is deprecated.) | HBase-1.0.x (Hadoop 1.x is NOT supported) | HBase-1.1.x | HBase-1.2.x | HBase-1.3.x | HBase-2.0.x |
|---|---|---|---|---|---|---|---|
| Hadoop-1.0.x | X | X | X | X | X | X | X |
| Hadoop-1.1.x | S | NT | X | X | X | X | X |
| Hadoop-0.23.x | S | X | X | X | X | X | X |
| Hadoop-2.0.x-alpha | NT | X | X | X | X | X | X |
| Hadoop-2.1.0-beta | NT | X | X | X | X | X | X |
| Hadoop-2.2.0 | NT | S | NT | NT | X | X | X |
| Hadoop-2.3.x | NT | S | NT | NT | X | X | X |
| Hadoop-2.4.x | NT | S | S | S | S | S | X |
| Hadoop-2.5.x | NT | S | S | S | S | S | X |
| Hadoop-2.6.0 | X | X | X | X | X | X | X |
| Hadoop-2.6.1+ | NT | NT | NT | NT | S | S | S |
| Hadoop-2.7.0 | X | X | X | X | X | X | X |
| Hadoop-2.7.1+ | NT | NT | NT | NT | S | S | S |

https://hbase.apache.org/book.html#hadoop

# HBase Versions



## § 4. Basic Prerequisites

This section lists required services and some required system configuration.

*Table 2. Java*

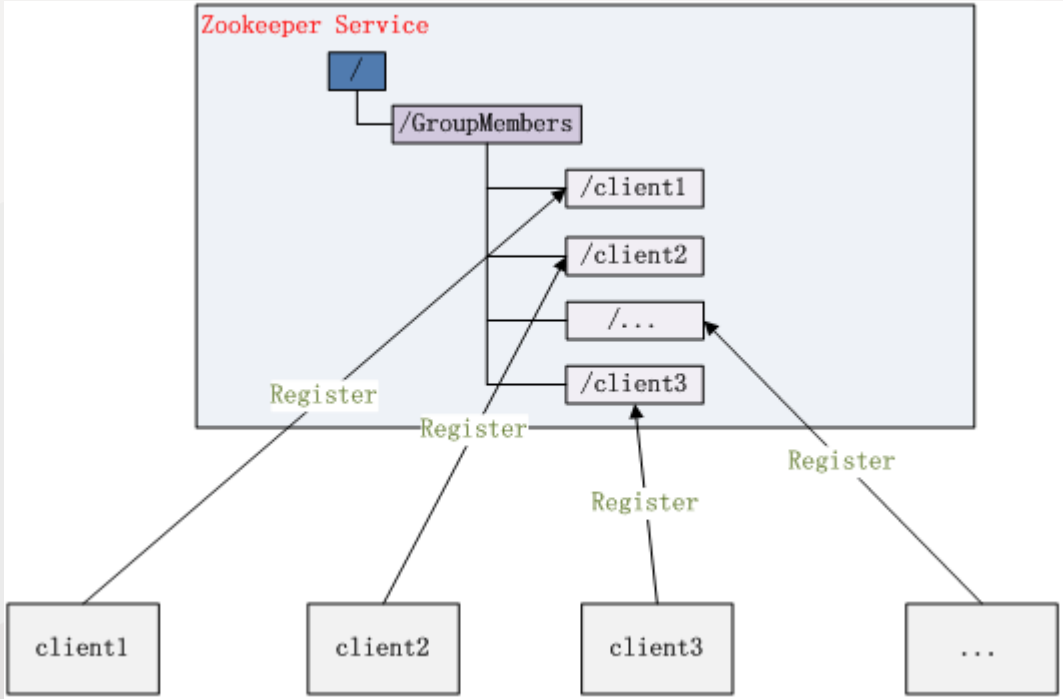
| HBase Version | JDK 6 | JDK 7 | JDK 8 |
|---|---|---|---|
| 2.0 | Not Supported | Not Supported | yes |
| 1.3 | Not Supported | yes | yes |
| 1.2 | Not Supported | yes | yes |
| 1.1 | Not Supported | yes | Running with JDK 8 will work but is not well tested. |
| 1.0 | Not Supported | yes | Running with JDK 8 will work but is not well tested. |
| 0.98 | yes | yes | Running with JDK 8 works but is not well tested. Building with JDK 8 would require removal of the deprecated `remove()` method of the `PoolMap` class and is under consideration. See HBASE-7608 for more information about JDK 8 support. |
| 0.94 | yes | yes | N/A |

In HBase 0.98.5 and newer, you must set `JAVA_HOME` on each node of your cluster. *hbase-env.sh* provides a handy mechanism to do this.

https://hbase.apache.org/book.html

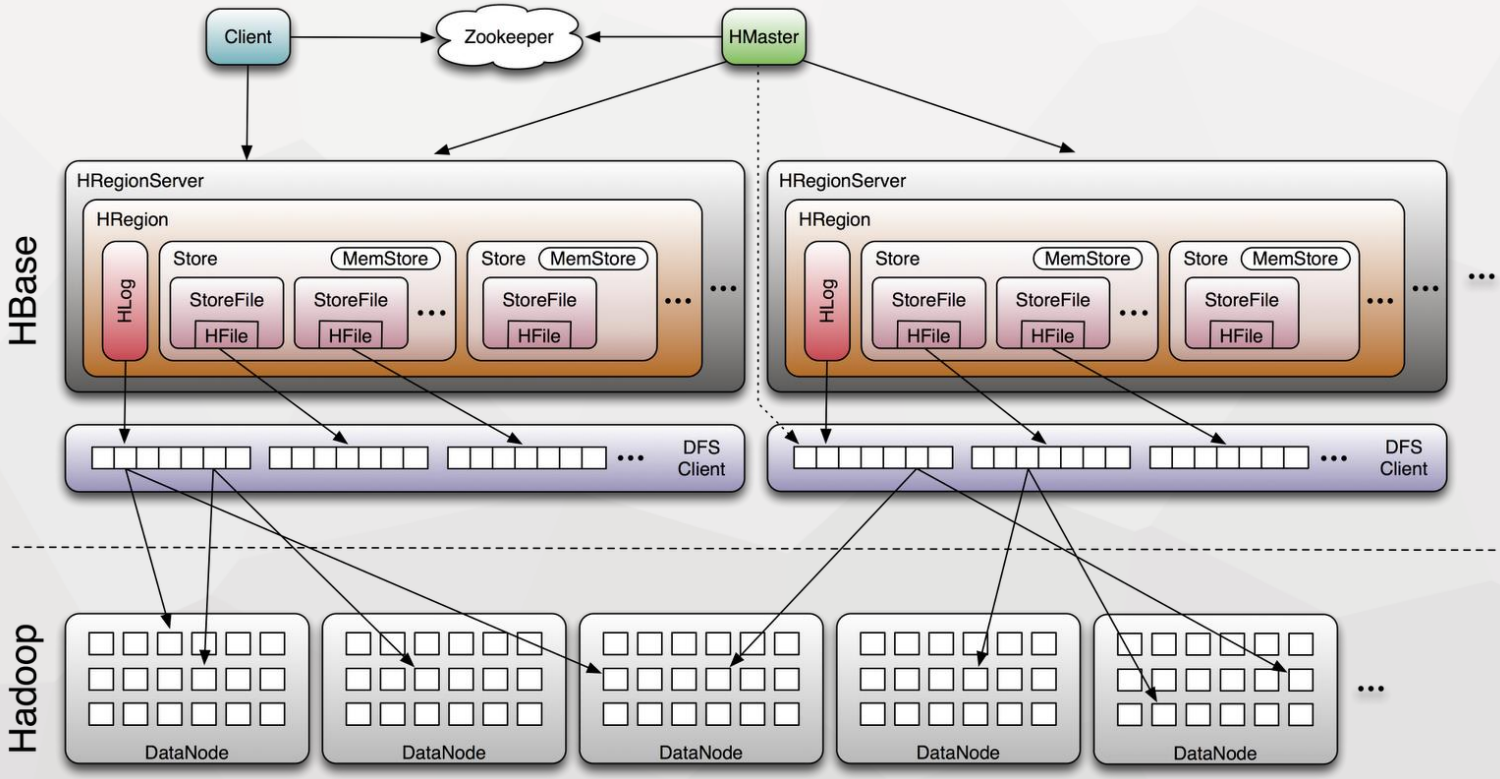
# architecture





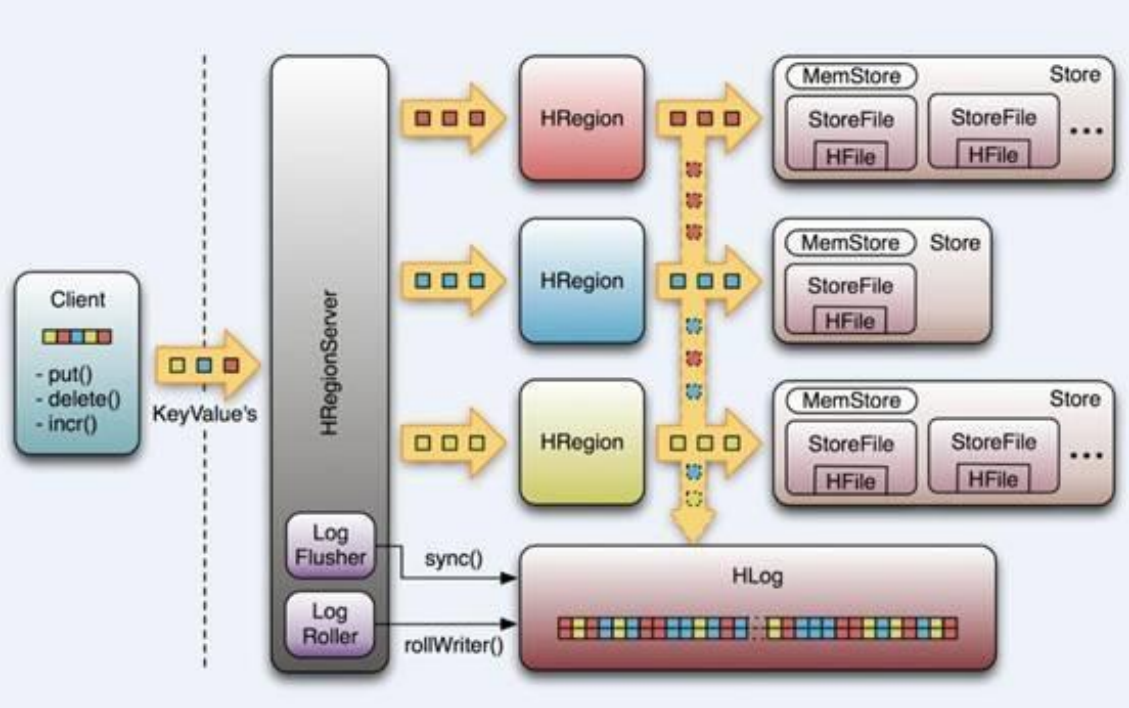
# architecture





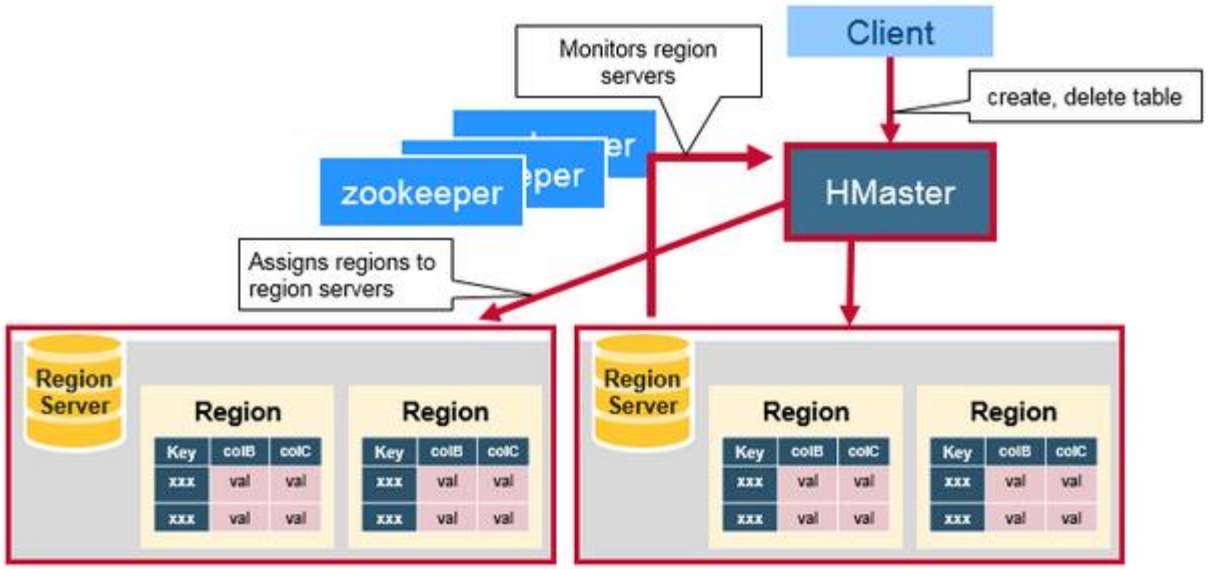
# architecture





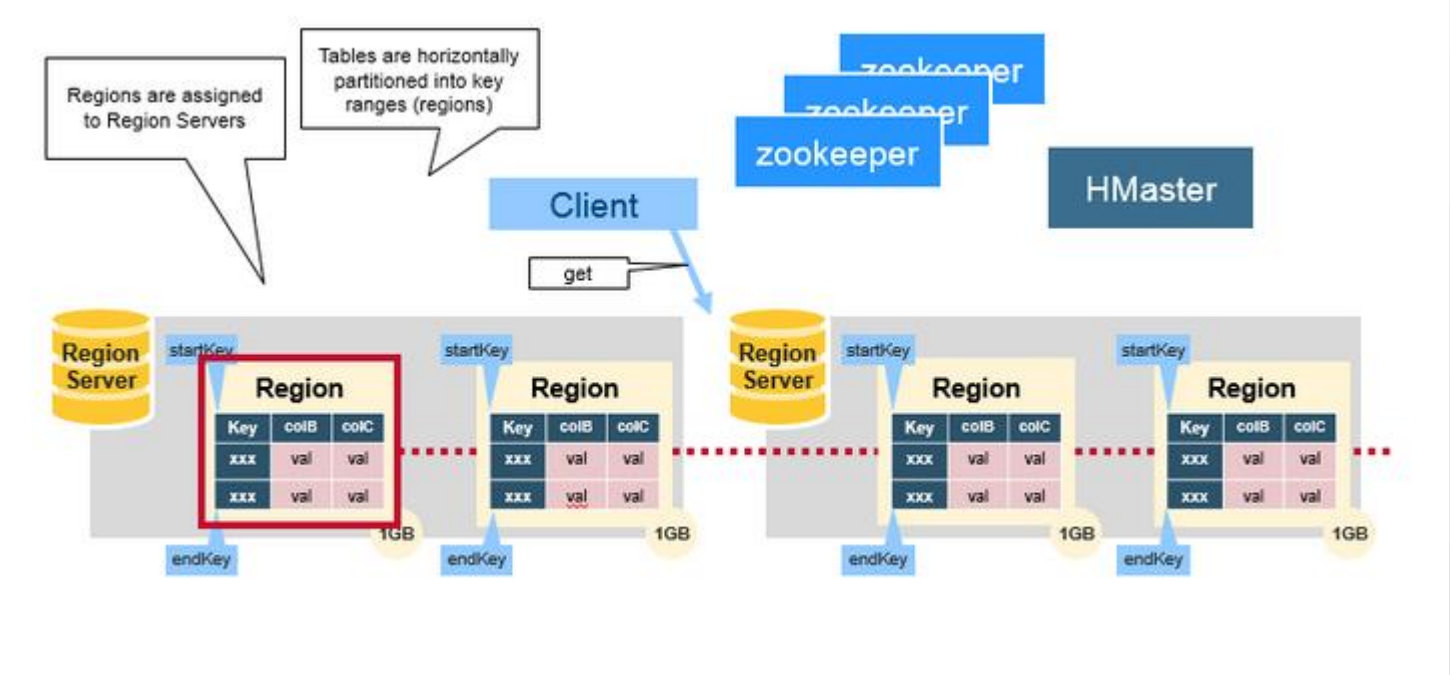
# architecture





# architecture





# PART3

## HBase Basic

# Column Family



"user_data" Column Family

"pictures" Column Family

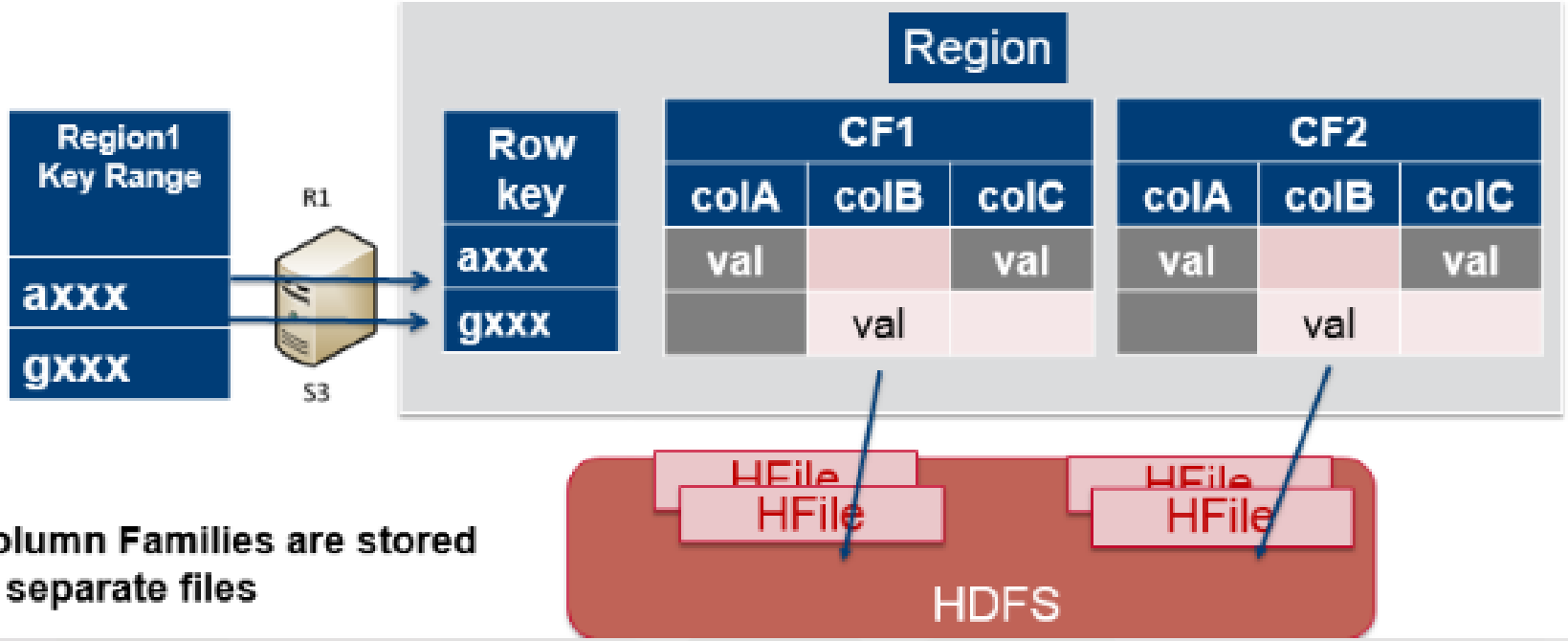
| | fname | lname | picture |
|---|---|---|---|
| aputrell@apache.org | "Andrew" | "Putrell" | |
| jdcryans@apache.org | "Jean-Daniel" | "Cryans" | downfall.jpg |
| stack@apache.org | | "Stack" | dancing_stack.jpg |
| todd@apache.org | "Todd" | "Lipcon" | turbo.jpg |

Row Key identifies a row across Column Families

Column Families store frequently accessed data together. Settings can be customized on a per Column Family basis

# Column Family






Column Families are stored in separate files

# Column Family



Column families

Column names

Rows

| | Contactinfo | | Prifileinfo |
|---|---|---|---|
| Rowkey | Fname | Lname | Image |
| jdupont | Jean | Dupont | |
| jsmith | John | Smith | <smith.jpg> |
| mrossi | Mario | Rossi | <mario.jpg> |

File

| | Contactinfo | |
|---|---|---|
| Rowkey | Fname | Lname |
| jdupont | Jean | Dupont |
| jsmith | John | Smith |
| mrossi | Mario | Rossi |

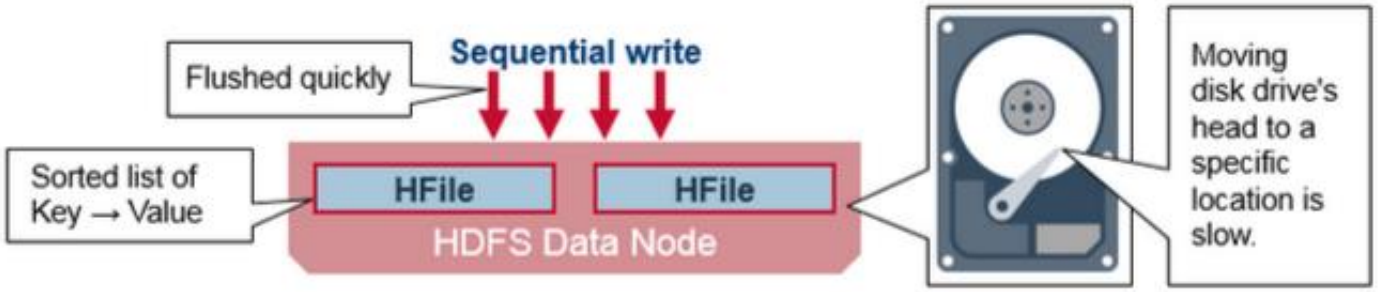
File

| | Prifileinfo |
|---|---|
| Rowkey | Image |
| jdupont | |
| jsmith | <smith.jpg> |
| mrossi | <mario.jpg> |

# Table



Rowkey + column + timestamp -> value




Key | Value

| Key | CF1:Col | version | value |
|---|---|---|---|
| ra | cf1:ca | v1 | 1 |
| rb | cf1:cb | v2 | 4 |
| rb | cf1:cb | v1 | 3 |
| rc | cf1:ca | v1 | 5 |

Key | Value

| Key | CF2:Col | version | value |
|---|---|---|---|
| ra | cf2:ca | v1 | 2 |
| rc | cf2:ca | v2 | 7 |
| rc | cf2:ca | v1 | 6 |
| rc | cf2:cd | v1 | 8 |

# Table



Rowkey + column + timestamp -> value

| Rowkey | Column | Timestamp | Cell value |
|---|---|---|---|
| jdupont | Contactinfo:fname | 1273746289103 | Jean |
| jdupont | Contactinfo:lname | 1273878447049 | Dupont |
| jsmith | Contactinfo:fname | 1273516197868 | John |
| jsmith | Contactinfo:lname | 1273871824184 | Smith |
| mrossi | Contactinfo:fname | 1273616297446 | Mario |
| mrossi | Contactinfo:lname | 1273971921442 | Rossi |

# RDBMS VS HBase



| | RDBMS | HBase |
|---|---|---|
| **Data layout** | Row- or column-oriented | Column family-oriented |
| **Transactions** | Yes | Single row only |
| **Query language** | SQL | get/put/scan |
| **Security** | Authentication/ Authorization | Access control at per-cell level, also at cluster, table, or row level |
| **Indexes** | Yes | Row key only |
| **Max data size** | TBs | PB+ |
| **Read/write throughput limits** | 1000s queries/second | Millions of queries/second |

# HBase



General  HBase shell commands

status
Show cluster status. Can be ‘summary’, ‘simple’, or ‘detailed’. The default is ‘summary’.
hbase> status
hbase> status ‘simple’
hbase> status ‘summary’
hbase> status ‘detailed’

version
Output this HBase versionUsage:
hbase> version

whoami
Show the current hbase user.Usage:
hbase> whoami

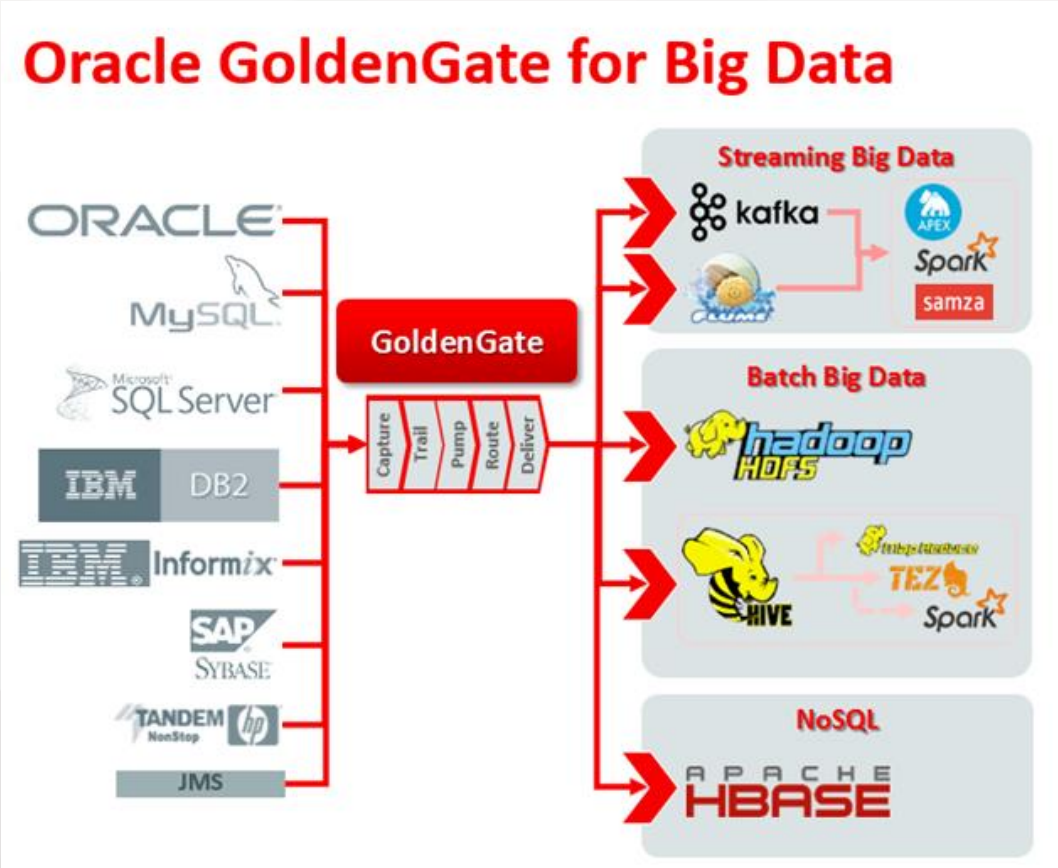
# RDBMS VS NoSQL



| | | Relational | | Non-Relational |
|---|---|---|---|---|
| Analytics | Proprietary Storage | Amazon Redshift, IBM Netezza, EMC Greenplum, Oracle, HP Vertica, Teradata — MPP | | |
| Analytics | Hadoop Storage | Cloudera Impala, Hive, Presto — SQL-on-Hadoop | | MapReduce |
| Operational | | Traditional SQL | NewSQL | NoSQL |
| Operational | Proprietary Storage | Oracle<br>DB2<br>SQL Server<br>MySQL | User-Sharded MySQL, NuoDB, Clustrix — On-Disk<br>MemSQL, VoltDB — In-Memory | Key Value: Aerospike, Riak<br>Column Family: Cassandra<br>Document: MongoDB<br>Graph: Neo4j, InfiniteGraph |
| Operational | Hadoop Storage | | Splice Machine — On-Hadoop | Column Family: HBase |

# Disparate Data Sources





# Write Heavy VS Read Heavy





## YCSBワークロードの種類

以下4種類を測定

Write Heavy

Read Heavy

| Workload | Application Example | Operation Ratio | Record Selection |
|---|---|---|---|
| Write-Only | Log | Read: 0%<br>Write: 100% | Zipfian(※) |
| Write-Heavy | Session Store | Read: 50%<br>Write: 50% | |
| Read-Heavy | Photo tagging | Read: 95%<br>Write: 5% | |
| Read-Only | Cache | Read: 100%<br>Write: 0% | |

(※) Zipfian分布: アクセス頻度が, 鮮度とは関係なく決まる
一部がヘッド / 大多数がテール

# Hbase Shell



```
hbase(main):002:0> create

ERROR: wrong number of arguments (0 for 1)

Here is some help for this command:
Creates a table. Pass a table name, and a set of column family
specifications (at least one), and, optionally, table configuration.
Column specification can be a simple string (name), or a dictionary
(dictionaries are described below in main help output), necessarily
including NAME attribute.
Examples:

Create a table with namespace=ns1 and table qualifier=t1
  hbase> create 'ns1:t1', {NAME => 'f1', VERSIONS => 5}

Create a table with namespace=default and table qualifier=t1
  hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}
  hbase> # The above in shorthand would be the following:
  hbase> create 't1', 'f1', 'f2', 'f3'
  hbase> create 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000, BLOCKCACHE => true}
  hbase> create 't1', {NAME => 'f1', CONFIGURATION => {'hbase.hstore.blockingStoreFiles' => '10'}}

Table configuration options can be put at the end.
Examples:

  hbase> create 'ns1:t1', 'f1', SPLITS => ['10', '20', '30', '40']
  hbase> create 't1', 'f1', SPLITS => ['10', '20', '30', '40']
  hbase> create 't1', 'f1', SPLITS_FILE => 'splits.txt', OWNER => 'johndoe'
  hbase> create 't1', {NAME => 'f1', VERSIONS => 5}, METADATA => { 'mykey' => 'myvalue' }
  hbase> # Optionally pre-split the table into NUMREGIONS, using
  hbase> # SPLITALGO ("HexStringSplit", "UniformSplit" or classname)
  hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO => 'HexStringSplit'}
  hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO => 'HexStringSplit', CONFIGURATION => {'hbase.hregion.scan.loadColumnFamilie
sOnDemand' => 'true'}}
```

https://hbase.apache.org/book.html#shell

# Java API

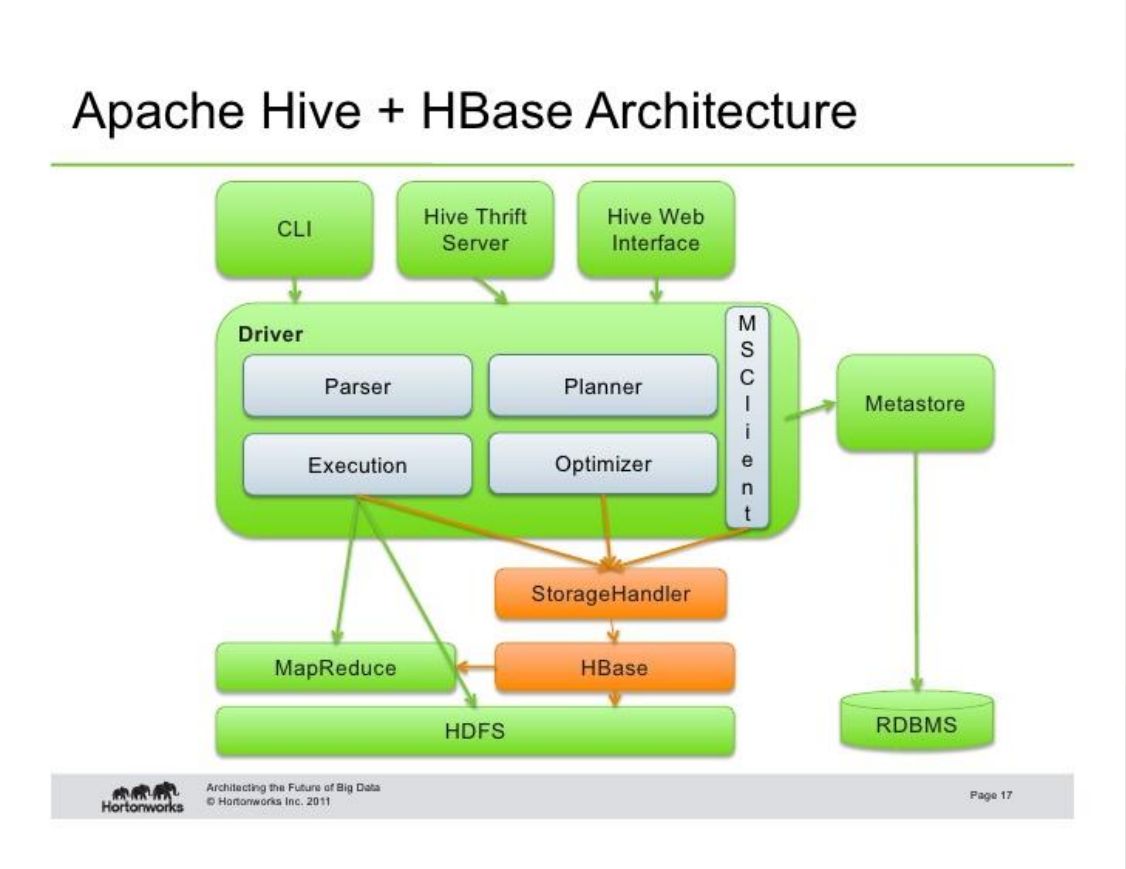


```java
public class HBaseExample {
  public static void main(String[] args) throws Exception {
    AbstractHBaseDBO dbo = new HBaseDBOImpl();

    //*drop if table is already exist.*
    if(dbo.isTableExist("user")){
     dbo.deleteTable("user");
    }

    //*create table*
    dbo.createTableIfNotExist("user",HBaseOrder.DESC,"account");
    //dbo.createTableIfNotExist("user",HBaseOrder.ASC,"account");

    //create index.
    String[] cols={"id","name"};
    dbo.addIndexExistingTable("user","account",cols);

    //insert
    InsertQuery insert = dbo.createInsertQuery("user");
    UserBean bean = new UserBean();
    bean.setFamily("account");
    bean.setAge(20);
    bean.setEmail("ncanis@gmail.com");
    bean.setId("ncanis");
    bean.setName("ncanis");
    bean.setPassword("1111");
    insert.insert(bean);

    //select 1 row
    SelectQuery select = dbo.createSelectQuery("user");
    UserBean resultBean = (UserBean)select.select(bean.getRow(),UserBean.class);

    // select column value.
    String value = (String)select.selectColumn(bean.getRow(),"account","id",String.class);

    // search with option (QSearch has EQUAL, NOT_EQUAL, LIKE)
    // select id,password,name,email from account where id='ncanis' limit startRow,20
    HBaseParam param = new HBaseParam();
    param.setPage(bean.getRow(),20);
    param.addColumn("id","password","name","email");
    param.addSearchOption("id","ncanis",QSearch.EQUAL);
    select.search("account", param, UserBean.class);

    // search column value is existing.
    boolean isExist = select.existColumnValue("account","id","ncanis".getBytes());

    // update password.
    UpdateQuery update = dbo.createUpdateQuery("user");
    Hashtable<String, byte[]> colsTable = new Hashtable<String, byte[]>();
    colsTable.put("password","2222".getBytes());
    update.update(bean.getRow(),"account",colsTable);

    //delete
    DeleteQuery delete = dbo.createDeleteQuery("user");
    delete.deleteRow(resultBean.getRow());
```

https://hbase.apache.org/book.html#jdo
https://hbase.apache.org/apidocs/index.html

# Thrift / RESTful





# PART4

## HBase Environment

# Docker




Master
Slave
Master
Slave
Master
Slave
HBase-Base
Spark-Base
Hadoop-Base
DNS
OS

# Web UI



HBASE | Home | Local Logs | Log Level | Debug Dump | Metrics Dump | HBase Configuration

## Server Metrics

Base Stats | Memory | Requests | hlogs | Storefiles | Queues | Block Cache

| Requests Per Second | Num. Regions | Block locality | Slow HLog Append Count |
|---|---|---|---|
| 5404 | 4 | 100 | 0 |

## Tasks

Show All Monitored Tasks | Show non-RPC Tasks | Show All RPC Handler Tasks | Show Active RPC Calls | Show Client Operations | View as JSON

No tasks currently running on this node.

## Block Cache

Base Info | Config | Stats | L1 | L2

| Attribute | Value | Description |
|---|---|---|
| Cache DATA on Read | true | True if DATA blocks are cached on read (INDEX & BLOOM blocks are always cached) |
| Cache DATA on Write | false | True if DATA blocks are cached on write. |
| Cache INDEX on Write | false | True if INDEX blocks are cached on write |
| Cache BLOOM on Write | false | True if BLOOM blocks are cached on write |
| Evict blocks on Close | false | True if blocks are evicted from cache when an HFile reader is closed |
| Compress blocks | false | True if blocks are compressed in cache |
| Prefetch on Open | false | True if blocks are prefetched into cache on open |

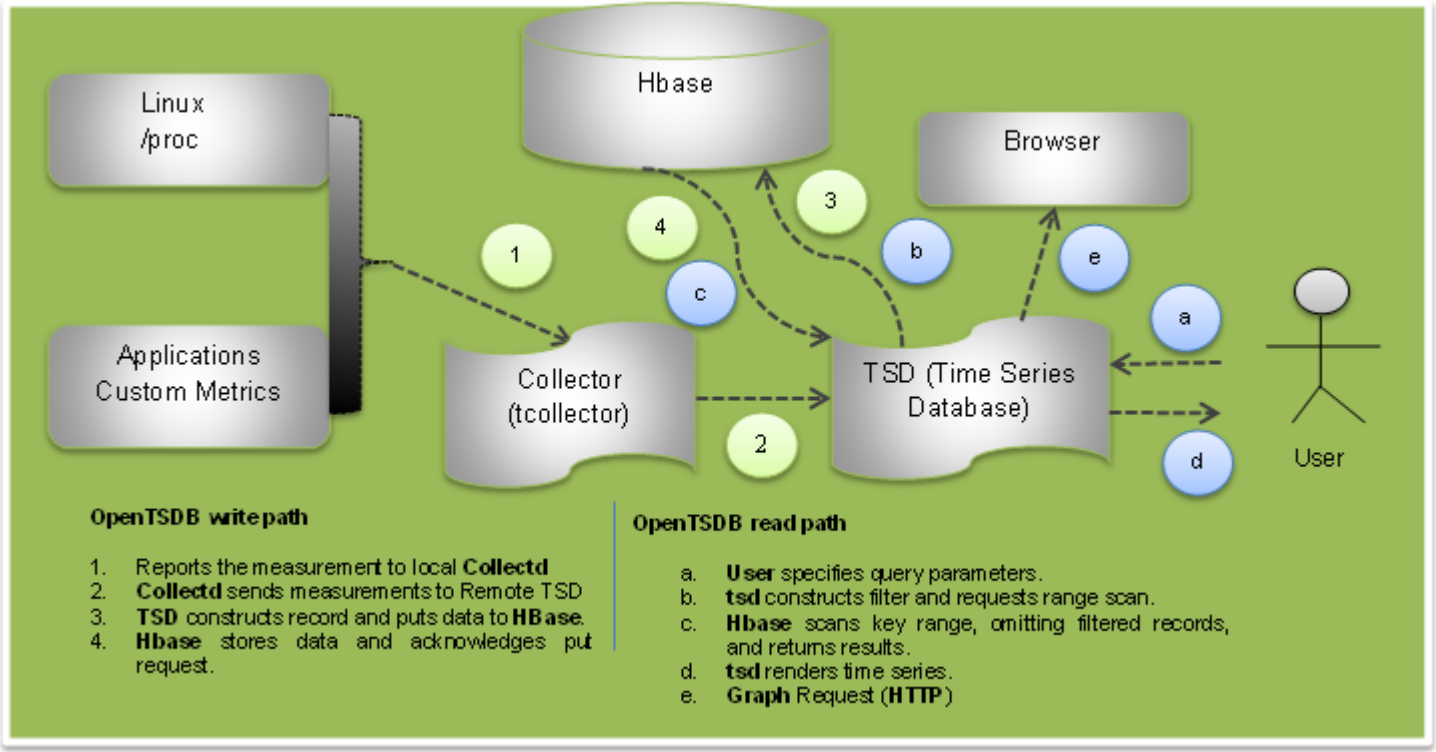
## Regions

# PART5

## Use Case

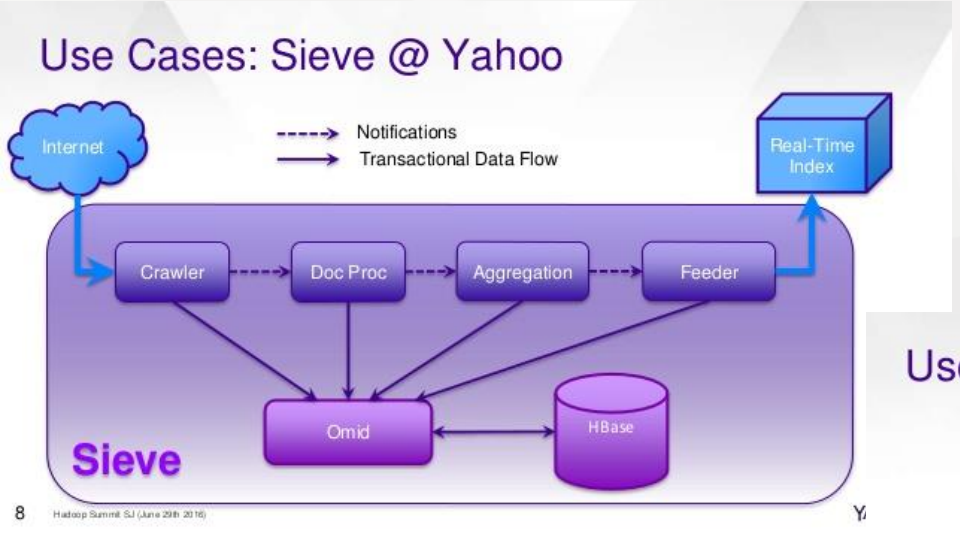
# Use Case

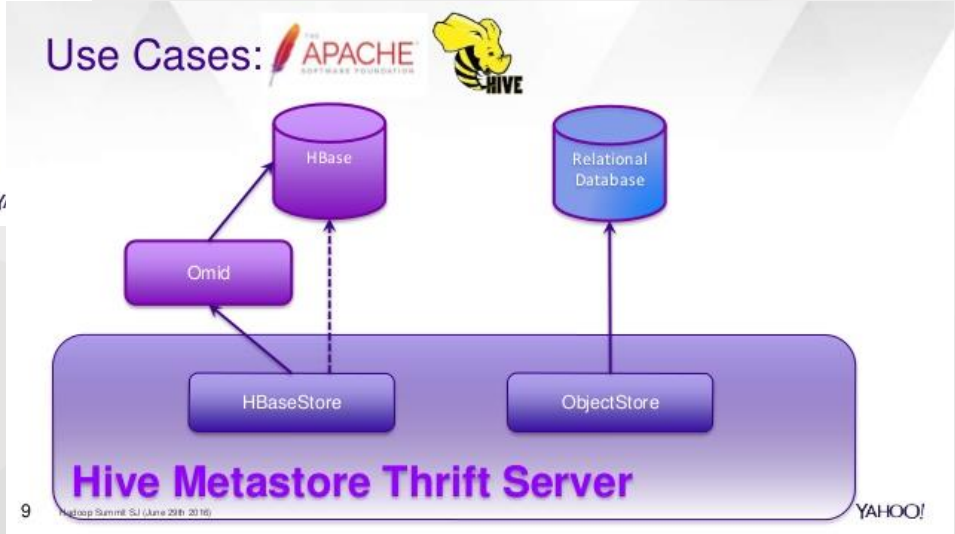



Hbase
Linux
/proc
Browser
Applications
Custom Metrics
Collector
(tcollector)
TSD (Time Series
Database)
User
1
2
3
4
a
b
c
d
e
OpenTSDB write path
1. Reports the measurement to local Collectd
2. Collectd sends measurements to Remote TSD
3. TSD constructs record and puts data to HBase.
4. Hbase stores data and acknowledges put request.
OpenTSDB read path
a. User specifies query parameters.
b. tsd constructs filter and requests range scan.
c. Hbase scans key range, omitting filtered records, and returns results.
d. tsd renders time series.
e. Graph Request (HTTP)

# Use Case




Use Cases: Sieve @ Yahoo
Internet
Notifications
Transactional Data Flow
Real-Time Index
Crawler
Doc Proc
Aggregation
Feeder
Omid
HBase
Sieve
8
Hadoop Summit SJ (June 29th 2016)


Use Cases:
APACHE
HIVE
HBase
Relational Database
Omid
HBaseStore
ObjectStore
Hive Metastore Thrift Server
9
Hadoop Summit SJ (June 29th 2016)
YAHOO!

# PART6

## Quick Start

# Quick Start



```
hbase(main):001:0> help
HBase Shell, version 0.91.0-SNAPSHOT, r1130916, Sat Jul 23 12:44:34 CEST 2011
Type 'help "COMMAND"', (e.g. 'help "get"' -- the quotes are necessary) for
help on a specific command. Commands are grouped. Type 'help "COMMAND_GROUP"',
(e.g. 'help "general"') for help on a command group.

COMMAND GROUPS:
  Group name: general
  Commands: status, version

  Group name: ddl
  Commands: alter, create, describe, disable, drop, enable, exists,
            is_disabled, is_enabled, list
```

# Quick Start



```
hbase(main):0xx:0>status
hbase(main):0xx:0>version
hbase(main):0xx:0>create 'member','member_id','address','info'
hbase(main):0xx:0>list
hbase(main):0xx:0>describe 'member‘
```

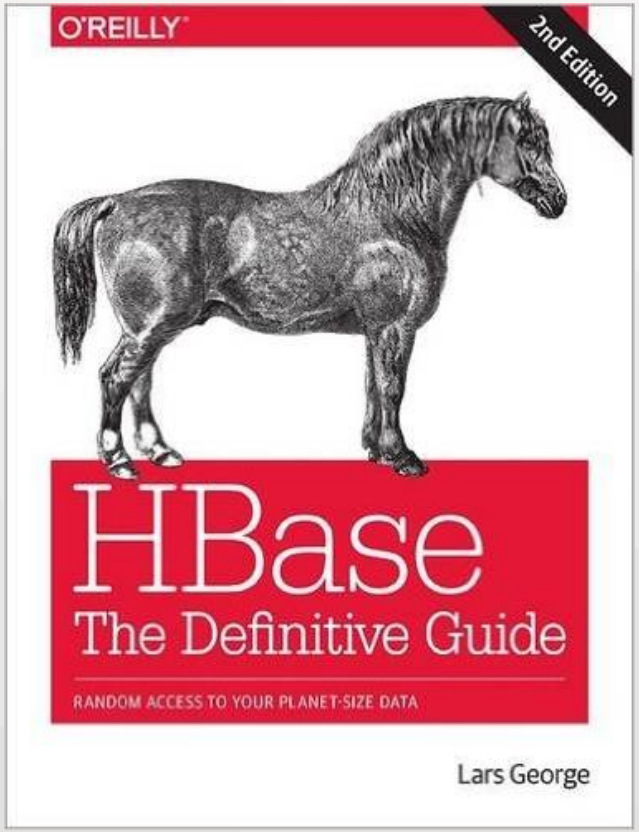
https://hbase.apache.org/book.html

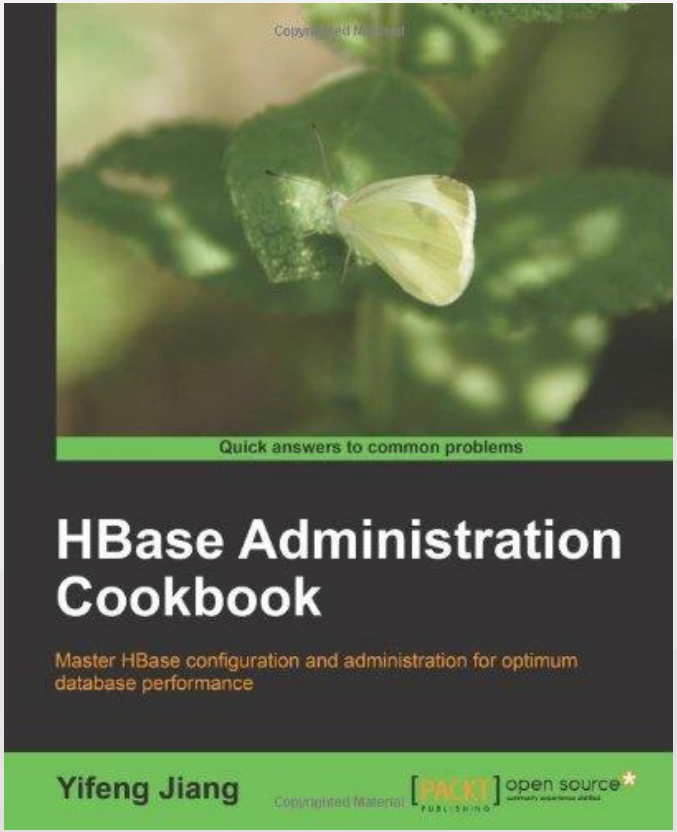
# PART7

## Reference Books

# Books




O'REILLY
2nd Edition
HBase
The Definitive Guide
RANDOM ACCESS TO YOUR PLANET-SIZE DATA
Lars George


Quick answers to common problems
HBase Administration Cookbook
Master HBase configuration and administration for optimum database performance
Yifeng Jiang
PACKT
open source

# The End