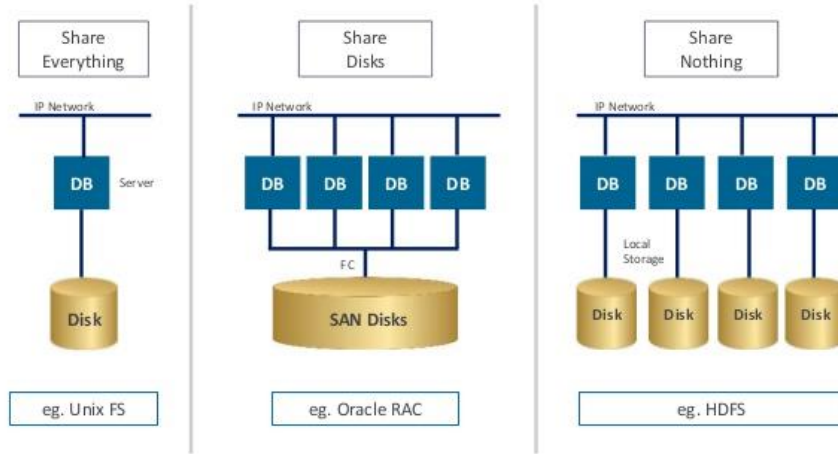


SHARE NOTHING ARCHITECTURE



Hbase Cluster

2017.4 XenRon

CONTENTS

Row Key

01

Schema Design

02

API

03

Cluster & HA

04

05

Backup & Restore





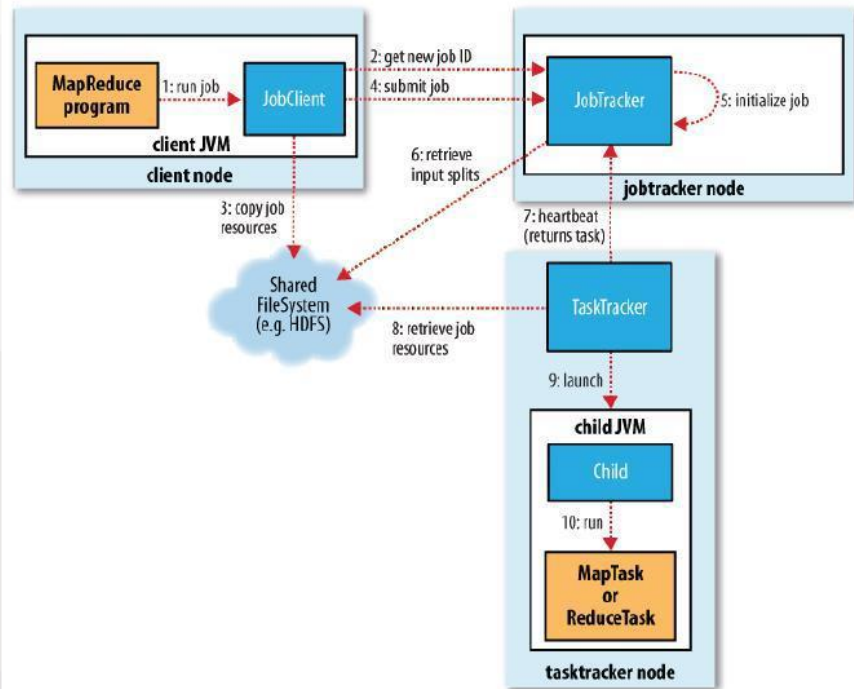
Review



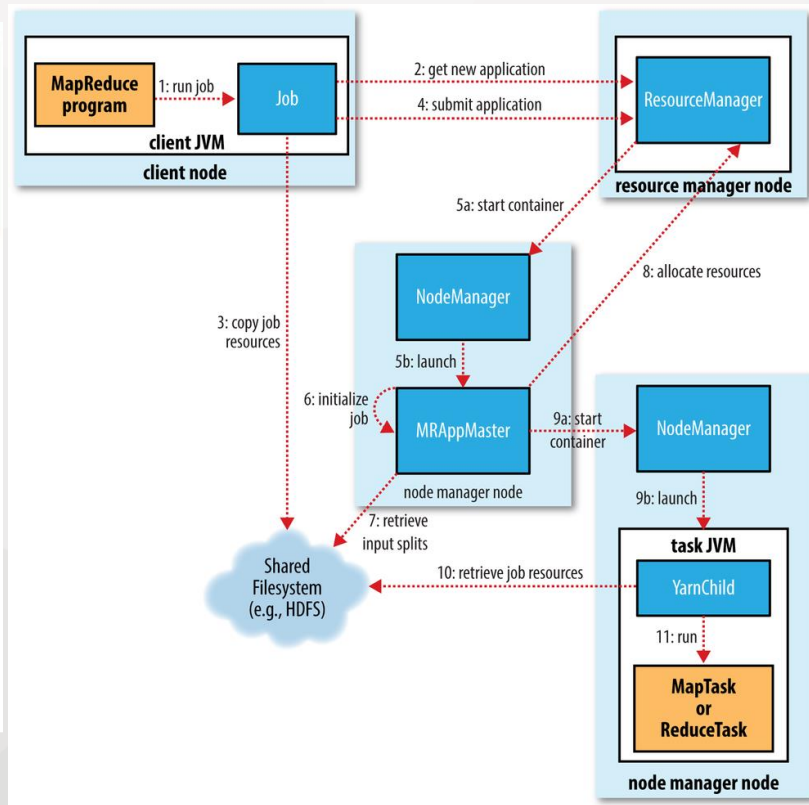
Review

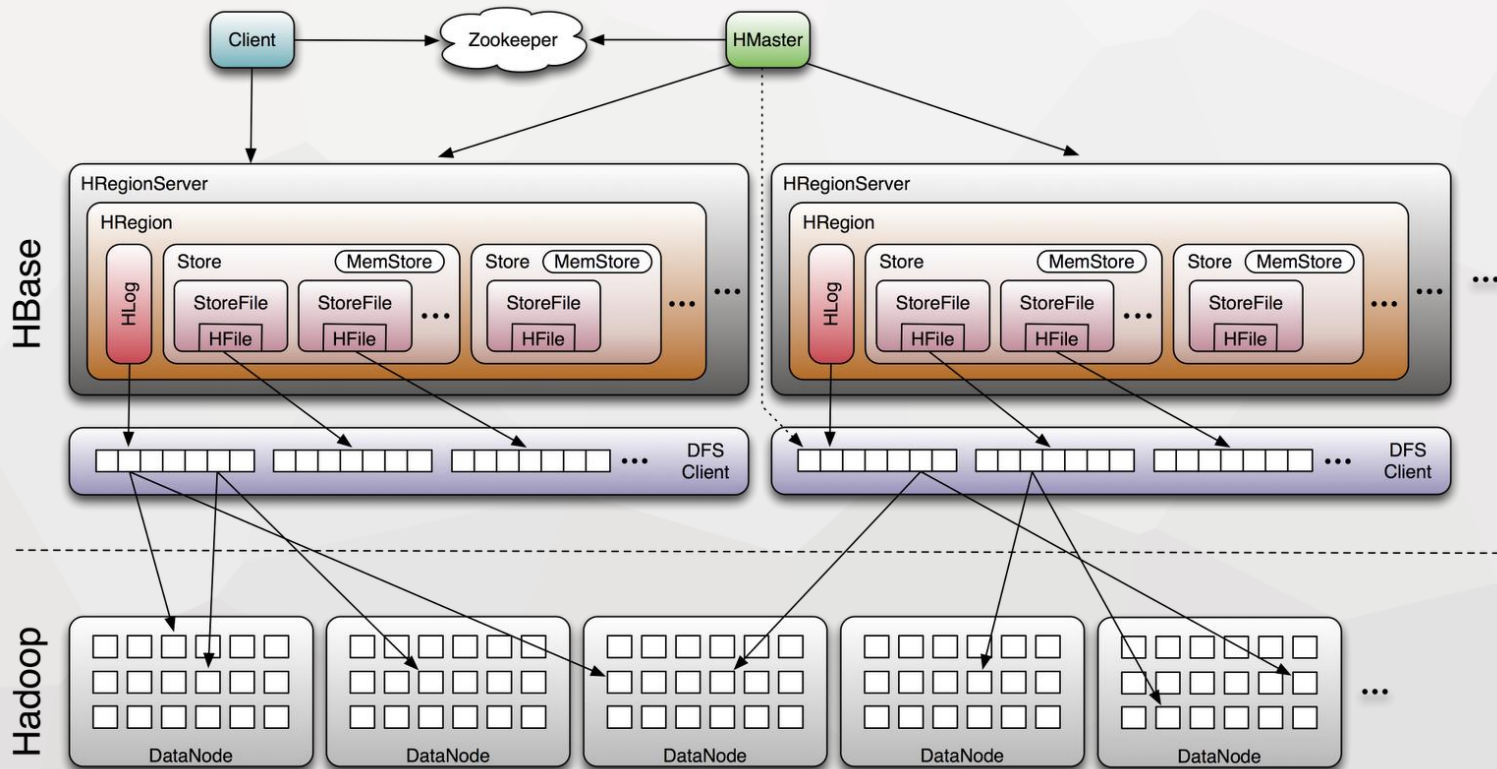
Review

Hadoop 1.x



Hadoop 2.x







PART1

Row Key Type

Sequential Keys

```
<timestamp><more key>: {CF: {CQ: {TS : Val}}}
```

- Hotspotting on Regions: **bad!**
- Instead do one of the following:
 - Salting
 - Prefix <timestamp> with distributed value
 - Binning or bucketing rows across regions
 - Key field swap/promotion
 - Move <more key> before the timestamp (see OpenTSDB later)
 - Randomization
 - Move <timestamp> out of key

Row Key Type

Original Row Key:

000000000000

000000000001

000000000002

000000000003

000000000004

000000000005

000000000006

000000000007



Salted Row Key:

0:000000000000

1:000000000001

2:000000000002

3:000000000003

4:000000000004

5:000000000005

6:000000000006

7:000000000007

Original Row Key:

000000000000:775

000000000001:314

000000000002:314

000000000003:310

000000000004:916

000000000005:925

000000000006:775



Promoted Row Key:

775:000000000000

314:000000000001

314:000000000002

310:000000000003

916:000000000004

925:000000000005

775:000000000006

Original Row Key:

000000000000
000000000001
000000000002
000000000003
000000000004
000000000005
000000000006



Random MD5 Hashed Row Key:

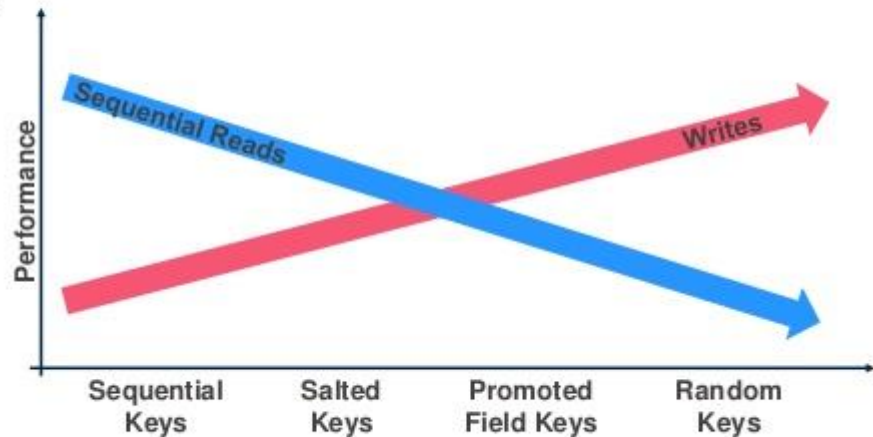
645a8aca5a5b84527c57ee2f153f1946
d67f0826d4c0aa7e3ea5861616a822b2
c93c5cedf7fba468e0fe2c845837abc7
6a1ae0e285acaf40dc30d13b702e6470
e57ea6134fc5278023292f1941dff865
63b307e583982c0746a5617e94f12dca
0d51268ce5ae7eed7e1ccd6d3859d033

Row Key Type

11

Sequential vs. Random keys

Random is better for writing , but sequential is better for scanning row keys



<https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Put.html>
<https://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/Get.html>



PART2

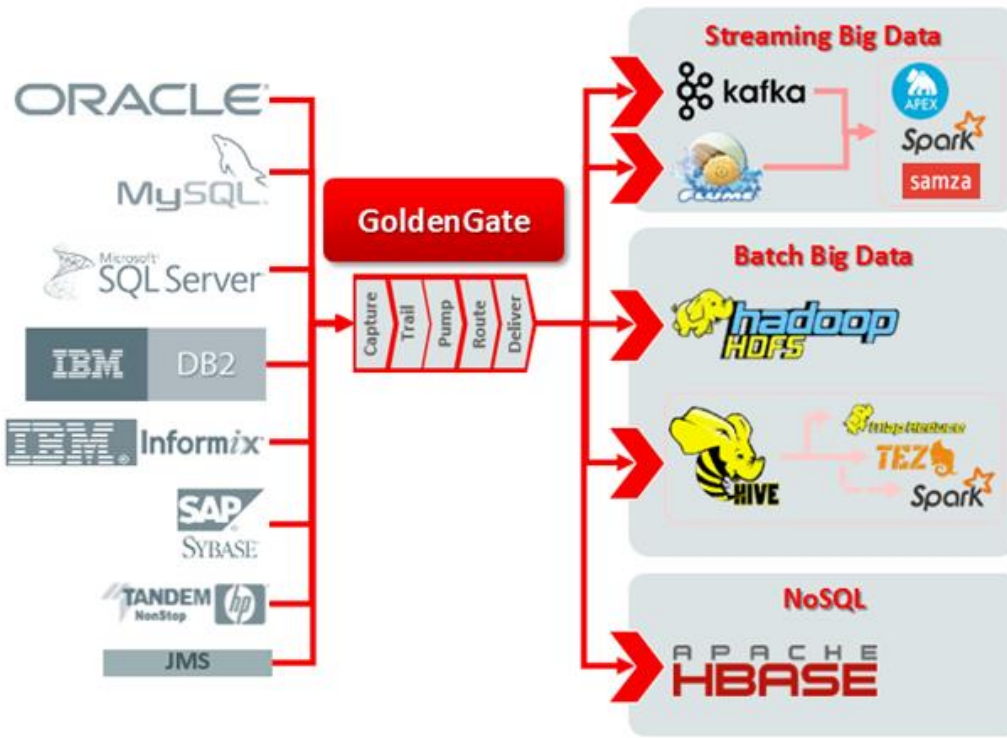


Schema Design

RDBMS VS NoSQL

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

Oracle GoldenGate for Big Data



Write Heavy VS Read Heavy

15

+ YCSBワークロードの種類

以下4種類を測定



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

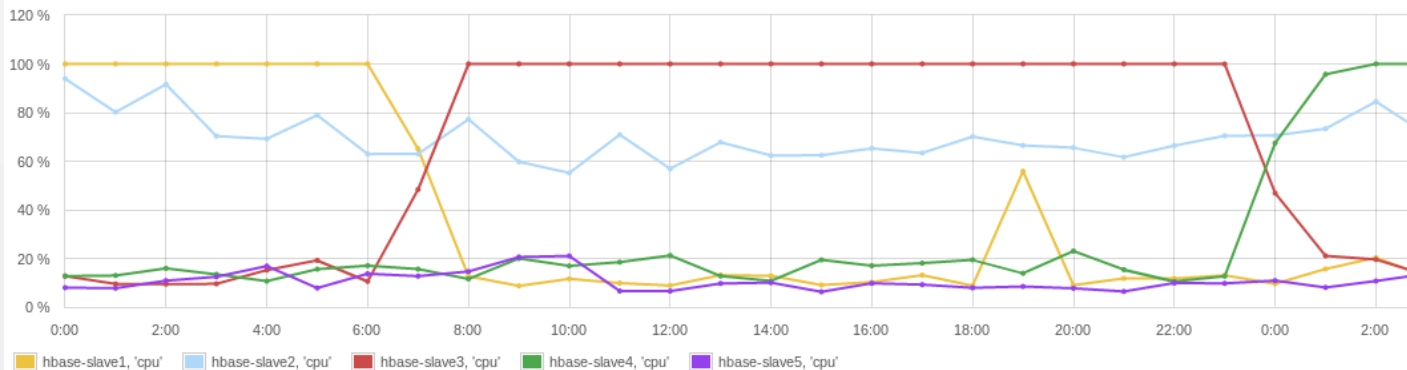
(※) Zipfian分布: アクセス頻度が、鮮度とは関係なく決まる
一部がヘッド / 大多数がテール

25

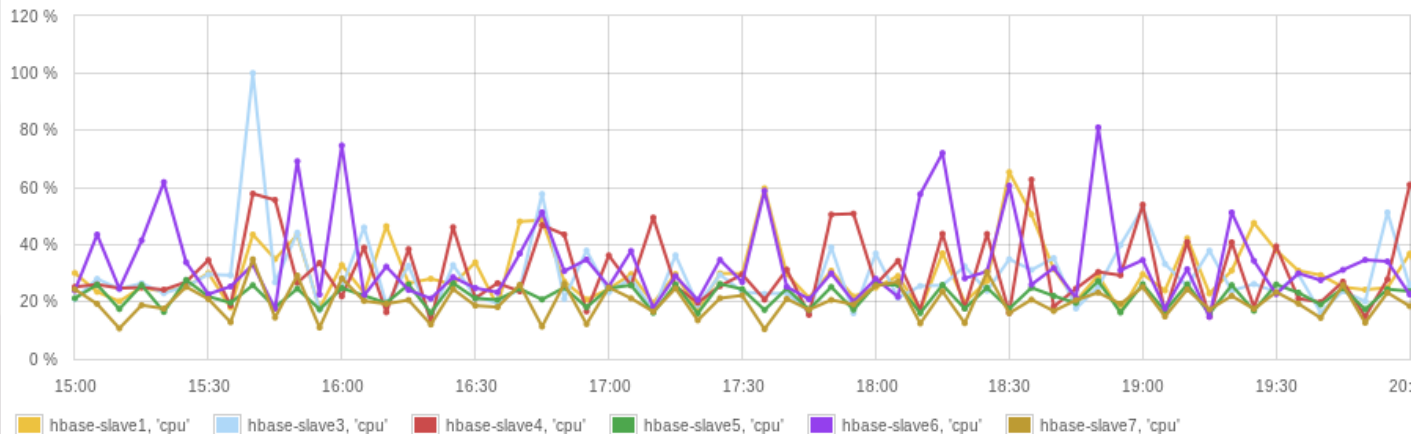
Hotspotting

16

Display metric : ☒ cpu ☐ memory



Display metric : ☒ cpu ☐ memory





PART3



API

```
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.loadColumnFamiliesOnDemand' => 'true'}}
```

```
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);

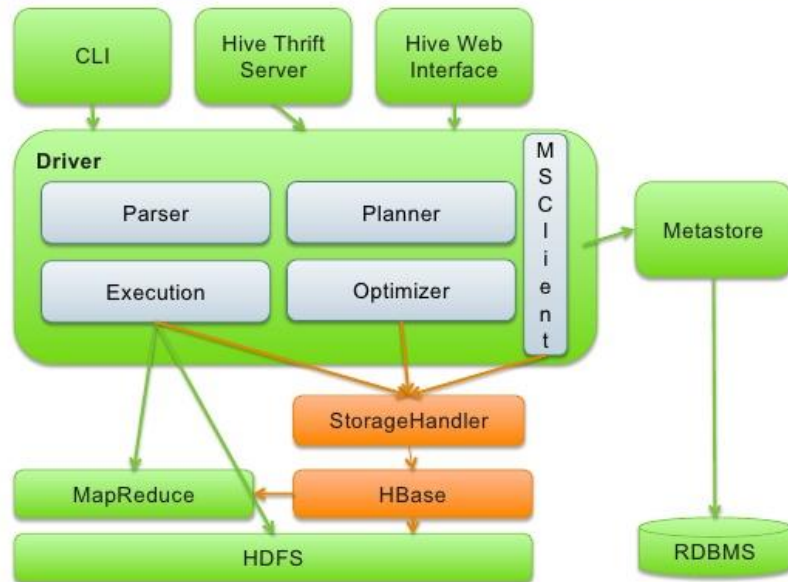
// search with option (QSearch has EQUAL, NOT_EQUAL, LIKE)
// select id,password,name,email from account where id='ncanis' limit startRow,20
HBaseParam param = new HBaseParam();
param.setPage(bean.getRow(),20);
param.addColumn("id","password","name","email");
param.addSearchOption("id","ncanis",QSearch.EQUAL);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

Apache Hive + HBase Architecture





PART4



Cluster & HA

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities										
Datanode Information										
In operation										
Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
sht-sgmhadoopdn-02.telenav.cn:50010 (172.16.101.59:50010)	1	In Service	31.25 GB	138.88 KB	12.74 GB	18.51 GB	0	138.88 KB (0%)	0	2.7.2
sht-sgmhadoopdn-03.telenav.cn:50010 (172.16.101.60:50010)	2	In Service	31.25 GB	557.73 MB	8.45 GB	22.25 GB	15	557.73 MB (1.74%)	0	2.7.2
sht-sgmhadoopdn-01.telenav.cn:50010 (172.16.101.58:50010)	0	In Service	31.25 GB	557.74 MB	11.43 GB	19.28 GB	15	557.74 MB (1.74%)	0	2.7.2
sht-sgmhadoopdn-04.telenav.cn:50010 (172.16.101.66:50010)	0	Decomission In Progress	31.25 GB	557.73 MB	12.44 GB	18.26 GB	15	557.73 MB (1.74%)	0	2.7.2
Decomissioning										
Node	Last contact	Under replicated blocks		Blocks with no live replicas		Under Replicated Blocks In files under construction				
sht-sgmhadoopdn-04.telenav.cn:50010 (172.16.101.66:50010)		15		0		0				

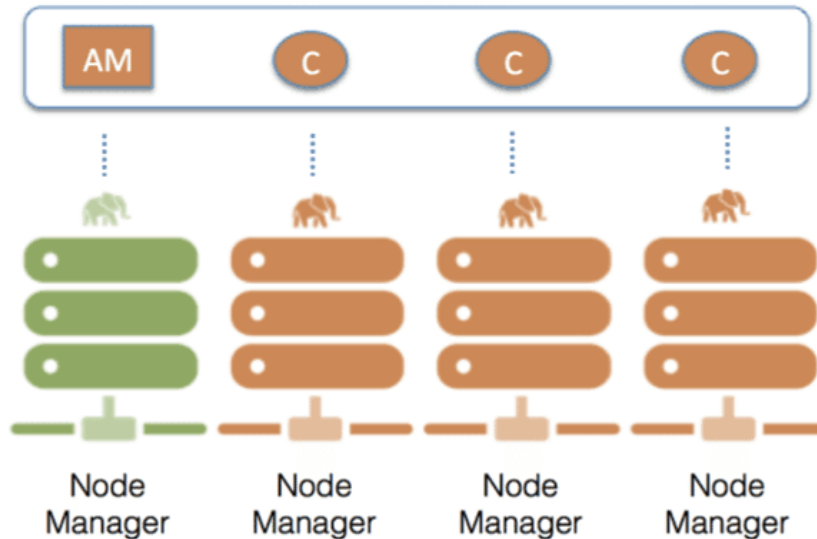
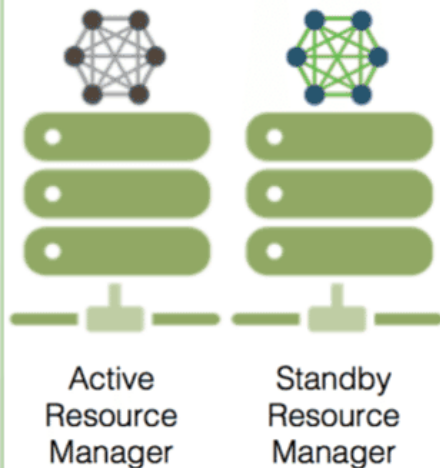
Upgrading a YARN cluster in a rolling fashion



New version



Old version





PART5



Backup & Restore

Backup & Restore

25

	Performance Impact	Data Footprint	Downtime	Incremental Backups	Ease of Implementation	Mean Time To Recovery (MTTR)
Snapshots	Minimal	Tiny	Brief (Only on Restore)	No	Easy	Seconds
Replication	Minimal	Large	None	Intrinsic	Medium	Seconds
Export	High	Large	None	Yes	Easy	High
CopyTable	High	Large	None	Yes	Easy	High
API	Medium	Large	None	Yes	Difficult	Up to you
Manual	N/A	Large	Long	No	Medium	High



The End