6/03/10 HBASE

```
$ hbase shell ↵

hbase(main):001:0> list ↵
 TABLE
     0 row(s)
hbase(main):002:0> create 'tab2', 'cf'
> describe 'tab2' ↵
      cf is column familiy
> list 'tablename'
     show particular table information

> put '~~row~~' '~~tab2~~', 'cf:a', ~~3~~100 ↵
      tab2 ,  row1 ,
> put 'tab2', 'row4', 'cf:b', 1000 ↵
> put 'tab2', 'row1', 'cf:b', 1000 ↵
> put 'tab2', 'row2', 'cf:a', 10 ↵
> put 'tab2', 'row2', 'cf:x', 1000 ↵
> scan 'tab2' ↵

  Row                        Column_cell
  row1                         —
  row1                         —
  row1                         —
  row2                         —
  row2                         —
```

```
> Create 'hr', 'emp', 'dept', 'mngrinfo'
> describe 'hr'

> put 'hr', 'emp1', 'emp:ecode', 'a101'
> put 'hr', 'emp1', 'emp:esal', 2000
> put 'hr', 'emp1', 'dept:dname', 'it'
> put 'hr', 'emp1', 'dept:dloc', 'hyd'
> scan 'hr'

> get 'hr', 'emp2'
```

(RDBMS)
TABLE HAS PRIMARY KEY:

```
Sqoop import -connect jdbc:mysql://localhost/mydb. --username
root --table emp -hbase-table hr1 --column-family emp
```

The primary key of RDBMS table is directed to row key of hbase table, Remaining Columns of RDBMS table are directed to Column family of hbase table. The above process will be failed if hbase table is not existed.

To create hbase base table dynamically

```
Sqoop import -connect jd
                  -
                  -
                  -
            --hbase-create-table
```

the option hbase table will import data if table existed. if table not existed it will create.

IF TABLE DOESN'T HAVE PRIMARY KEY:-

```
Sqop Import --connect Jdbc:mysql://localhost/mydb
--username _
--pwd _
--table Staff -m 1 --hbase-table hr3 -hbase-row-key ecode
-column-family emp --hbase-create-table
```

MYSQL: Tab

primarykey K a b c d

Requirement:

a, b Columns to be assigned to cf1 Column family

c, d columns to be assigned to cf2 Column family.

Step1:
```
$ Sqoop import --connect Jdbc:mysql://localhost/mydb
--username root
-- pwd --
-- table Tab --Columns a,b,K, --hbase-table Test
-- Column-family cf1 -hbase-create-table
```

Step2:
```
$ Sqoop import -Connect Jdbc:mysql//localhost/mydb
--username root
--table tab --Column K,c,d --hbase-table Test --column-
family cf2 --hbase-create-table
```

12/03/13

HBASE + hive integration :

```
hive> create table hbaseImage(ecode String, ename String, sex String,
 sal int, dname String, dloc string)
 > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
 > WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key emp.name,
   emp:sex, emp:sal, dept:dname, dept:dloc")
 > TBLPROPERTIES ("hbase.table.name"= "htab1");
```

mandatory

mapping:

hive	hbase
ecode	rowkey
name	emp:name
sex	emp:sex
sal	emp:sal
dname	dept:dname
dloc	dept:dloc

one to one mapping is done.

STORED BY : SERDE class name : which serializes and deserializes hbase format to hive table and hive format to hbase table and the class name is ~~hbase~~ HBaseStorageHandler which is availble in hbase-core.jar.

SERDE PROPERTIES: Columns mapping will be done

- * Columns of hive table and Hbase table mapping will be done
- * mapping Style is one to one mapping (1st Column to 1st Column, and so on)

Ex:- Image(name, age, ecode, city)
↳ key

they follow following
"hbase.Columns.mapping"
= emp : name, emp:age, :key, emp:city

order is mandatdy.

TBLPROPERTIES (optional) :

→ When hive table is created, HBase table will also be created with given name

ex: TBLPROPERTIES("hbase.table.name = "htab1").
if this option is not Specified, with Same name of hive table, hbase table will be created.

Integrating hive table with existed hbase table:

→ TBLPROPERTIES ("_", "_") ↳ give the name of existed table.

Copying hdfs file data into HBase table:

* Files delimiter is 'comma'

ex:- 101, AA, 2000
102, BB, 3000
103, CC, 3000

```
1) hive> create table stage (ecode string, ename string, esal int)
         row format delimited fields terminated by ',';

2) hive> load data inpath 'file1.txt' into table stage;
3) hive> Insert overwrite table Image1
         Select ecode, ename, esal from stage;
```

to find total salary of hbase table

```
hive> Select sum(sal) from Image1;
```

This query is processed by hbase.

Send Hbase table Aggregated results into DB2 table of target Reporting system.

HBASE DATA SCHEMA:

HBASE Table name: mytab

data schema

Row	cell
101	emp.name = Anirudth
101	emp.sal = 10000
101	dept: dname = IT
101	dept: dloc = hyd
102	emp. name = Avinash
102	emp.sal = 20000
102	dept:dloc = hyd
102	dept: dname = HR
103	emp: name = Ankit
103	emp: sal = 30000
103	dept: dname = HR
103	dept: dloc = pune

Step1: hive> Create Table empimage(ecode String, name String, sal int, dname String, dloc String) STORED By "org. apache. Hadoop. hive. hbase. HBaseStorageHandler" WITH SERDEPROPERTIES ("hbase. Column. Mapping" = ":key, emp:name, emp:sal, dept:dname, dept:dloc") TBLPROPERTIES ("hbase.table.name" = "mytab"); ↵

Step2: hive> create table Reshbase (dloc string, totalsal int);

Step3: hive> Insert overwrite table Reshbase
Select dloc, sum(sal) from empImage group by dloc;

Step4:

```
$ Sqoop export --connect jdbc:DB2://ds.myorg.com/mydb
  --username Dev001 --password xyz123
  --table LocAggr --export-dir /user/hive/warehouse/Reshbase/
                                              00000_0
```

Qn

Send Hbase table Aggregated Results into DB2 table of target Reporting System.

Exporting HBase Table data (as it is) into RDBMS using SQOOP

```
$ Sqoop export --connect jdbc:mysql://localhost/mydb
--username root
-- table target
-- hbase-table mytab
-- column-family cf1 ↵
```

The row Key of hbase table is mapped with primary key of table.

If RDBMS is not have primary key

ex:- you want to send row key to ecode(not primary key)

if --hbase-table mytab --hbase-row-key ecode
↓
Column of RDBMS.

NOTE: While exporting -m 1 is not required.

Limitations of hive and hbase integrations:-

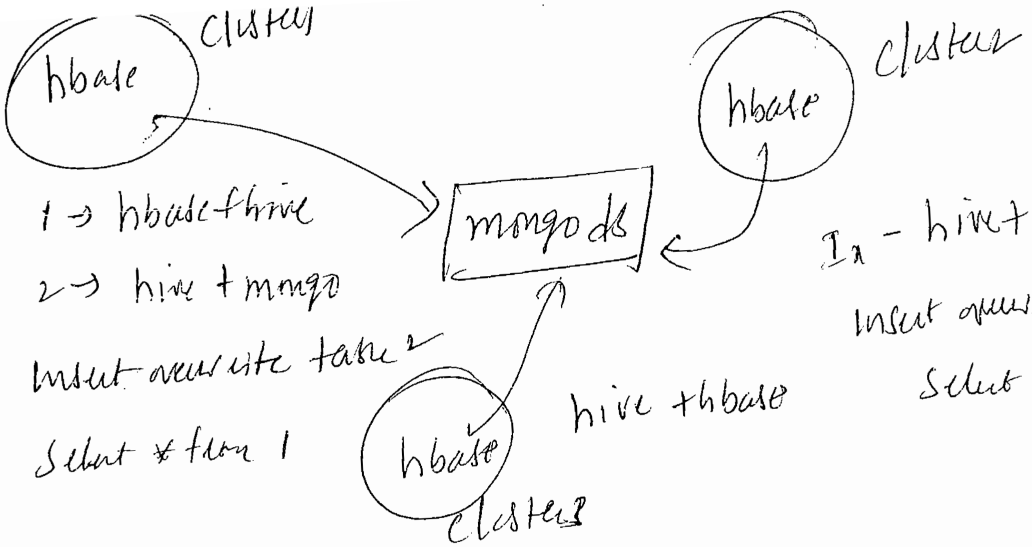
1) not possible for Remote hbase servers

2) We are not interact with Remote hbase Config with local hbase

Solution for this is convert into HDFS they interact with Remote Server.

hive is able to integrate with Remote mongo DB.

Ex: Create table Imagemongo (-, -, -) STORED BY 'org.apache.hadoop.hive.mongo.mongoloader' with serdeproperties ("_":"_")
Collection ("mongo1")
IPAddress ("190:390:700:8104")



1 -> hbase+hive
2 -> hive + mongo
Insert overwrite table 2
Select * from 1

I_x - hive+Mongo
Insert overwrite table I_x
Select * from I_x

Currently hive integration is available with

document { 1) mongodb
2) Couch db

(key,value) { 3) Riak
4) PNUTS(not open source)

graphs { 5) green plum
6) Neo4J

both local and Remote integration

7) HBASE 8) Cassandra

not Remote integration

HADOOP CASCADING.

U Y

In mapReduce, running job will be failed in following Situations.

(i) if output path is already existed

(2) if any bad record is encountered

Solutions in CASCADING FOR ABOVE PROBLEMS:

cascading has SinkMode to control the file existence operations, Such as failed, overwrite, Reuse (appending)

SinkMode KEEP

SinkMode REPLACE

SinkMode UPDATE

KEEP : Job will be failed if output file path is already existed that means it won't ~~stop~~ ~~data~~ distrub data of file.

REPLACE: It overwrites into file if file (output path) is already existed

UPDATE :- it appends new data to the existing file.

In cascading files (I/o-files) are specified in the form of Taps. there are three types of Taps.

1) Source Tap

2) SinkTap

3) Trap Tap

Source Tap : To Specify input file path

Sink Tap : to Specify output file path

Trap Tap : A file to capture the bad data

CASCADING TERMINOLOGY :

Tuple : tuple is a data record

Field : piece of data ex: line, word --etc

Pipe : is any operation, to be performed on data

Tap : Indicate a file (I/O)

Scheme : Schema design (field structure)

Input Scheme : Input file structure (field list of input file)

Output Scheme : output file structure (field list of output file)

Types of Pipes (operations) :

each : To perform functions/Filters on each tuple, Similar to Foreach of pig

every : To perform Aggregation ~~of~~ on Grouped data

GroupBy : to group tuples based on fields

CoGroup : to Join multiple ~~taps~~ Similar to SQLJoins Such as inner/outer, left/right

Aggregator : to Specify Aggregator function.

Pipe Assembly: Assembling All operations togeather in a Sequence

PipeStream: the data which passes through a pipe is called pipe Stream

FlowConnector:- which Connects Sourcetap, Sinktap, assembly

Flow: is a dataflow execution process, which starts from head of the Assembly to tail of the Assembly.

Structure of cascading program:-

```
≡ } Imports of cascading classes.
public class MainClass
{
      public static void main(String args[]) {
      Source Scheme decleration;
      Sink Scheme decleration;
      Source tap;
      Sink tap;
   head of the Assembly
   __________ operation(pipe1)        Flow instance;
   _____ operation(pipe2)             execute flow;
   _____         3                    }
   ≡             4
                 5              }
 Flow Connector
Assign MainClass in Application Jar file
```

Cascading Classes:

Scheme → to define Source & Sink Schemas.

Pkg: Cascade.pipe.Scheme

ex:- Scheme SourceScheme = new TextLine(new Fields("line"));

Scheme SinkScheme = new TextLine(new Fields("word", "count"));

Fields - to Speify field List of file

Pkg:- Cascade pipe.Field

Tap: to define input file (Source tap) and output file (Sink tap)

Pkg: Cascade.pipe.Tap

Eg: Tap Source = new Hfs(SourceScheme, inputpath);

Tap Sink = new Hfs(SinkScheme, outputpath); SinkMode.REPLACE

↓

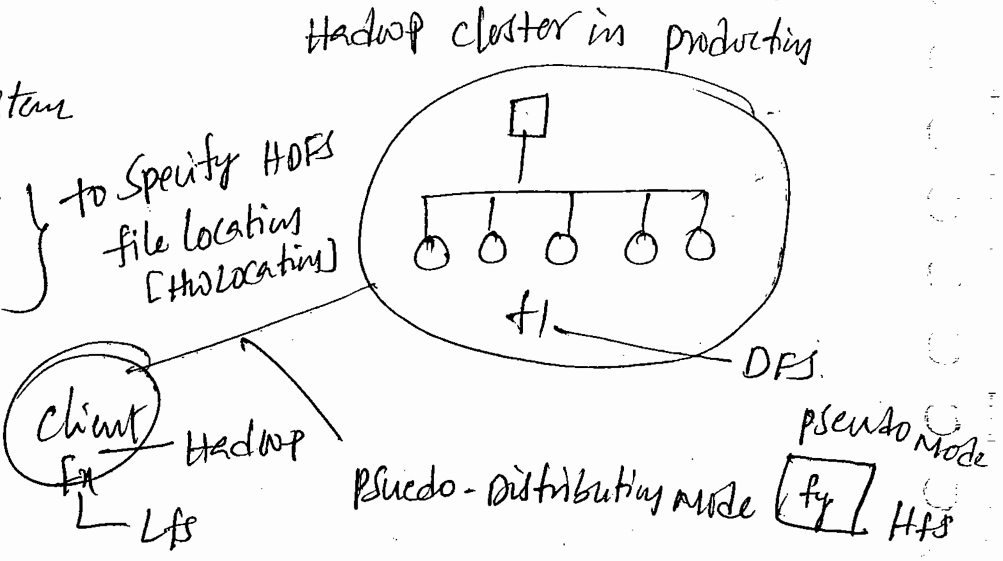
default SinkMode is SinkMode.KEEP

Hfs: Hadoop file system

Lfs: Local file system

Dfs: Distributed file System

Pkgs:- Cascade pipe.Hfs } to Specify HDFS file location [HDFS location]
Cascade pipe Lfs
cascade pipe DFS



HFS → Refers HDFS file of your psuedo-mode

LFS → Refers HDFS file & your client machine

DFS → Refers HDFS file of cluster (production)

Defining Assembly :- (Header)

→ Job name

Pipe assembly = new Pipe("wordCount");

pkg: cascade.pipe.Pipe

Pipe Operations:

define a function

String regex = "[^]*";

Function f = new RegexGenerator(new Fields("line"), regex);

(or)

Function f = new StringTokenizer(new Fields("line")).nextToken();

Applying Functions For each Tuple:

assembly = new Each(~~new Fields("line")~~, f);

assembly, new Fields("line"), f);

Grouping

assembly = new GroupBy(assembly, new Fields("word"));

Defining an Aggregator:-

Aggregator count = new ~~Aggregator~~ Count (new Fields(word), count);

Applying Aggregation on group:

assembly = new Every(assembly, count);

Flow Connector Instance:-

property p = new property();

FlowConnector.setApplicationJarClass(Mains.class);

FlowConnector fc = new FlowConnector(p);

Defining Flow:

Flow f = fc. connect (~~assembly, Source, Sink~~)
(Jobname (wordcount), Source, Sink, assembly);

executing flow:

f. complete();

Example program:

```
import Java. util. regex. RegexGenarator;
import Java. io. IOException;
import cascade. pipe. pipe;
import cascade. pipe. Hfs;
import cascade. pipe. scheme;
import cascade. pipe. Tap;
import cascade. pipe. Fields;
import cascade. pipe. Each;
import   "   " .Functions, GroupBy, Aggregator, FlowConnector, property,
                          Flow;
import org. apache. hadoop. util. GenericOptionparser;
 public class WordCount {
        public static void main( String str[]) {
              String[] files = new GenericOptionparser(args). getRemaining
                                                                   Args();
  Scheme Sourcescheme = new Textline (new Fields(~~"word", "count"~~ new Fields("line")));
  Scheme SinkSource = new Textline (new Fields ("word", "count"));
   Tap Source = new Hfs (SourceScheme, fields[0]);
```

Tap sink = new Hfs(SinkSource, files [1], SinkMode. REPLACE);

Pipe assembly = new Pipe(" word-count"); header of assembly

String regex = "[^]*";

Function func = new RegexGenerator (new Fields("line"), regex);

// Adding optional pipes to Assembly

assembly = new Each(assembly, new Fields("line"), func);

assembly = new GroupBy(assembly, new Fields("word"));

Aggregator count = new ~~Count~~(new Fields("word"));
COUNT

assembly = new Every(assembly, new Fields("count") count);

Properties p = new Properties();

Tail of Assembly

FlowConnector.setApplicationJarClass (wordCount. ~~jar~~ class);

FlowConnector fc = new FlowConnector (p);

Flow f = fc. ~~Count~~ connect(" word-count" source, sink, assembly
);

Source, Sink, assembly get together.

f. complete();

}

}

executing same as MR code