# APACHE HBASE

Xiaoming Gao

Hui Li

Thilina Gunarathne

INDIANA UNIVERSITY

## Outline

- HBase and Bigtable Storage
- HBase Use Cases
- HBase vs RDBMS
- Hands-on: Load CSV file to Hbase table with MapReduce

## Motivation

- Lots of Semi structured data
- Horizontal scalability
- Commodity hardware
- Tight integration with MapReduce
- RDBMS don’t scale
- Google BigTable (paper).

# Apache HBase

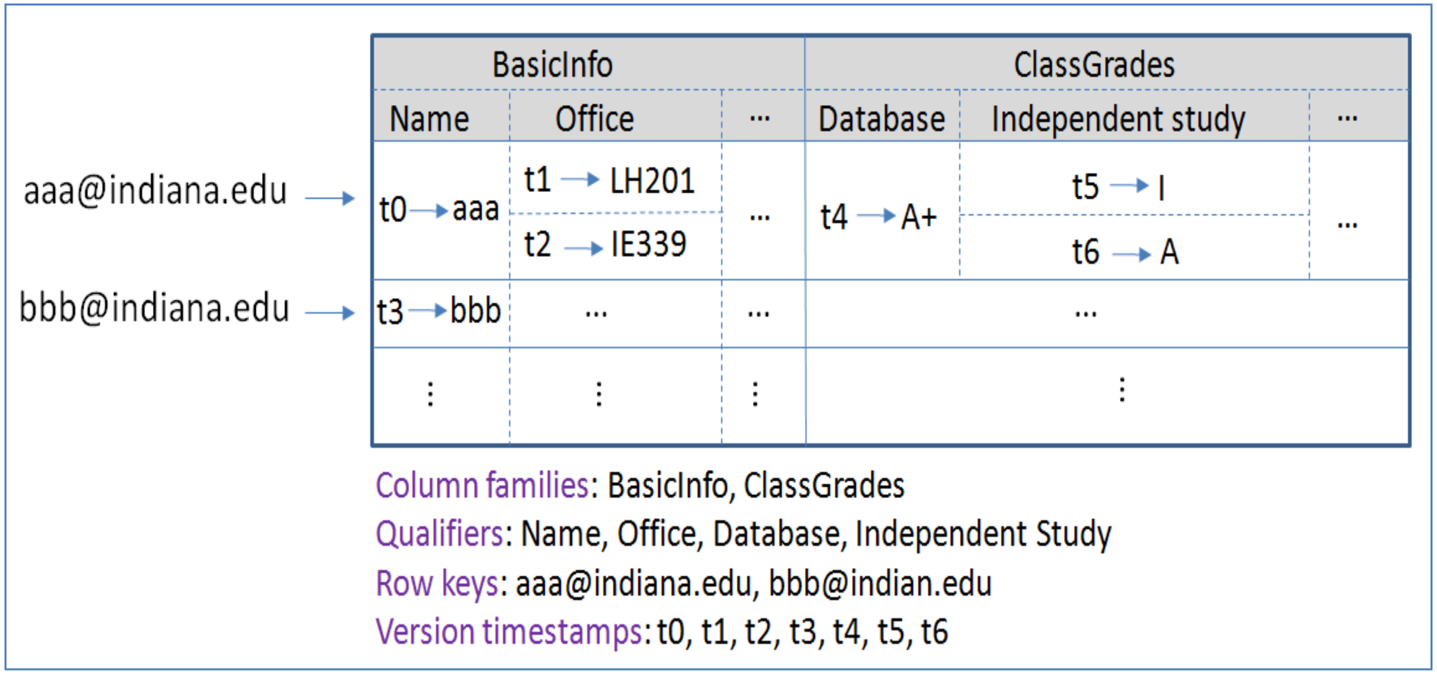
- Open-source, distributed, column-oriented, datastore.
- Hosting of very large tables atop clusters of commodity hardware.
- When you need random, realtime read/write access to your Big Data.
- Automatic partitioning
- Linear scalability

## HBase and Hadoop

- HBase is built on Hadoop
  - Fault tolerance
  - Scalability
- HBase uses HDFS for storage
  - Adds random read/write capability
- Batch processing with MapReduce

# Data Model: A Big Sorted Map

- A big sorted sparse Map
- Tables consist of rows, each of which has a primary key (row key)
- Each row can have any number of columns
- Multidimensional : map of maps
- Versioned

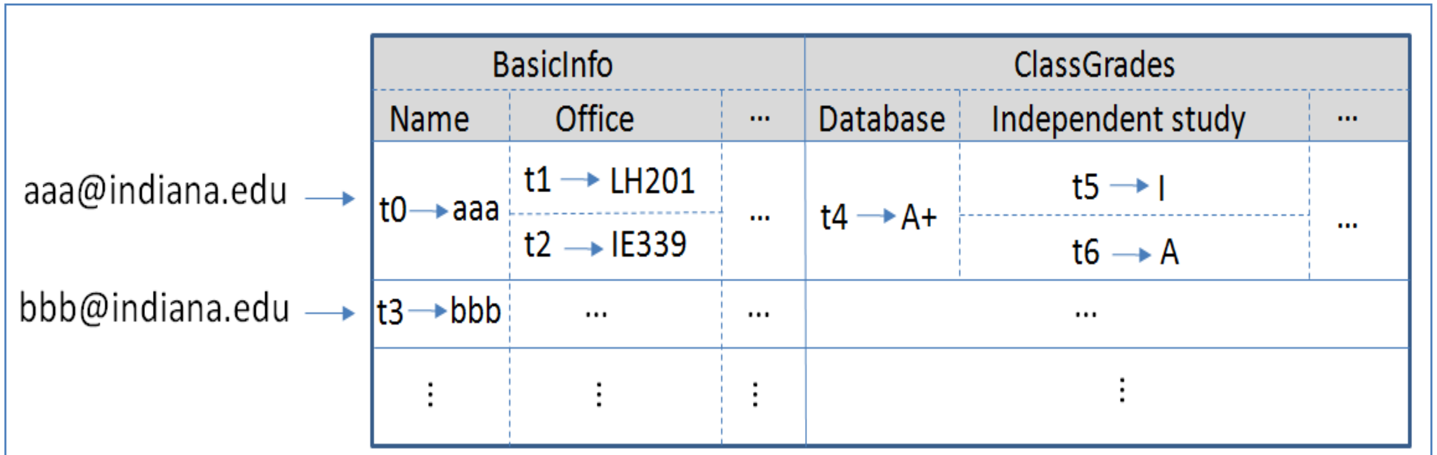


| | BasicInfo | | | ClassGrades | | |
|---|---|---|---|---|---|---|
| | Name | Office | … | Database | Independent study | … |
| aaa@indiana.edu → | t0 → aaa | t1 → LH201<br>t2 → IE339 | … | t4 → A+ | t5 → I<br>t6 → A | … |
| bbb@indiana.edu → | t3 → bbb | … | … | … | | |
| | ⋮ | ⋮ | ⋮ | ⋮ | | |

Column families: BasicInfo, ClassGrades
Qualifiers: Name, Office, Database, Independent Study
Row keys: aaa@indiana.edu, bbb@indian.edu
Version timestamps: t0, t1, t2, t3, t4, t5, t6

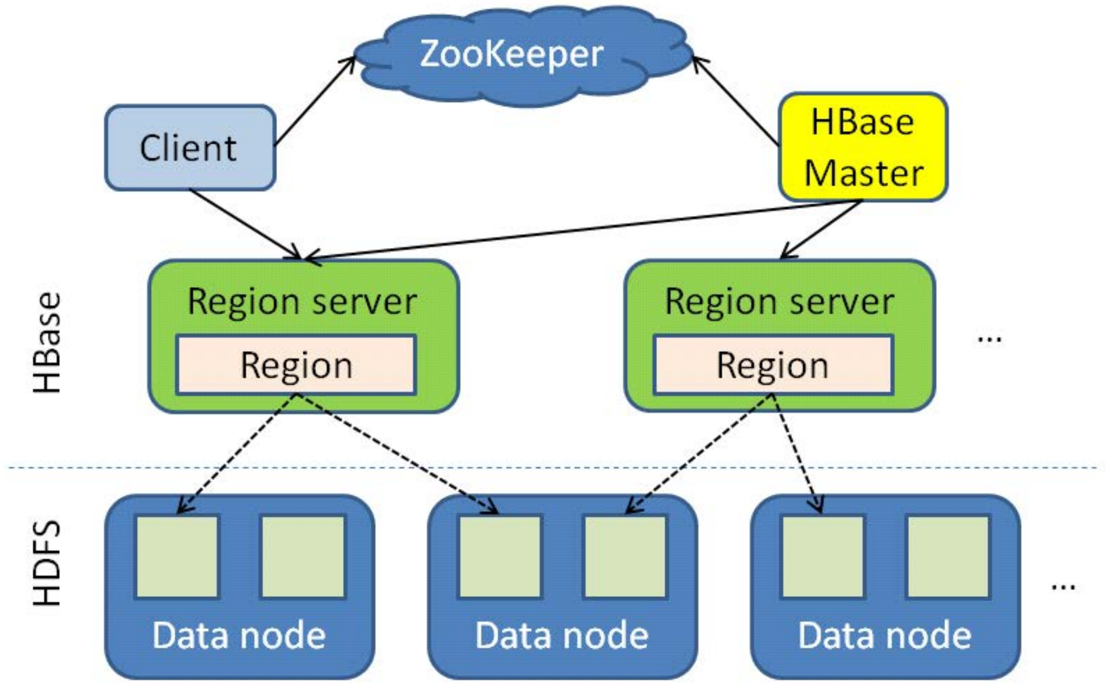
# Physical View

| RowKey | Column key | Time Stamp | value |
|---|---|---|---|
| aaa@indiana.edu | BasicInfo:Name | t0 | aaa |
| aaa@indiana.edu | BasicInfo:Office | t2 | IE339 |
| aaa@indiana.edu | BasicInfo:Office | t1 | LH201 |
| bbb@indiana.edu | BasicInfo:Name | t3 | bbb |
| ……… | | | |



Column families: BasicInfo, ClassGrades
Qualifiers: Name, Office, Database, Independent Study
Row keys: aaa@indiana.edu, bbb@indian.edu
Version timestamps: t0, t1, t2, t3, t4, t5, t6

# HBase Cluster Architecture



- Tables split into regions and served by region servers
- Regions vertically divided by column families into “stores”
- Stores saved as files on HDFS

# HBase VS. RDBMS

| | RDBMS | HBase |
|---|---|---|
| Data layout | Row-oriented | Column-family-oriented |
| Schema | Fixed | Flexible |
| Sparse data | Not | Good |
| Query language | SQL (Join, Group) | Get/Put/Scan  (& Hive) |
| Hardware requirement | Large arrays of fast and expensive disks | Designed for commodity hardware |
| Max data size | TBs | ~1PB |
| Read/write throughput | 1000s queries/second | Millions of queries/second |
| Easy of use | Relational data modeling, easy to learn | A sorted Map, significant learning curve, communities and tools are increasing |

## When to Use HBase

- Dataset Scale
  - Data sizes that cannot fit in a single node RDBMS
- High throughput
  - reads/writes are distributed as tables are distributed across nodes
- Batch Analysis
  - Massive and convoluted SQL queries can be executed in parallel via MapReduce jobs
- Large cache
- Sparse data
- Random read/write

## Use Cases:

- Facebook Analytics
  - Real-time counters of URLs shared, preferred links
- Twitter
  - 25 TB of message every month
- Mozilla
  - Store crashes report, 2.5 million per day.

## Programming with HBase

1. HBase shell
   - Scan, List, Create, Get, Put, Delete
2. Java API
   - Get, Put, Delete, Scan,…
3. Non-Java Clients
   - Thrift
   - REST
4. HBase MapReduce API
   - hbase.mapreduce.TableMapper;
   - hbase.mapreduce.TableReducer;
5. High Level Interface
   - Pig, Hive

INDIANA UNIVERSITY

# Hbase with Hadoop MapReduce

- Work with MapReduce
  - TableInputFormat & TableOutputFormat
  - Provides Mapper and Reducer base classes
  - Utility methods
    - TableMapReduceUtil
  - Writable types

# Hbase with Hadoop

```
Configuration config = HBaseConfiguration.create();
Job job = new Job(config,"ExampleSummary");
job.setJarByClass(MySummaryJob.class);     // class that contains mapper and reducer

Scan scan = new Scan();
scan.setCaching(500);        // 1 is the default , which will be bad for MapReduce jobs
scan.setCacheBlocks(false);  // don't set to true for MR jobs
……… // set other scan attrs

TableMapReduceUtil.initTableMapperJob(sourceTable,
                          scan,  MyMapper.class,  Text.class,  IntWritable.class,  job);
TableMapReduceUtil.initTableReducerJob(targetTable, MyTableReducer.class,   job);
job.setNumReduceTasks(1);

boolean b = job.waitForCompletion(true);
```

More info : [http://hbase.apache.org/book.html#mapreduce](http://hbase.apache.org/book.html#mapreduce)

## Hbase with Hadoop

```
public static class MyMapper extends TableMapper<Text, IntWritable>  {
        private final IntWritable ONE = new IntWritable(1);
        private Text text = new Text();

        public void map(ImmutableBytesWritable row,
                                        Result value, Context context) ..{
                String val = new String(
                                value.getValue(Bytes.toBytes("cf"), Bytes.toBytes("attr1")));
                text.set(val);
                context.write(text, ONE);
        }
}
```

More info : [http://hbase.apache.org/book.html#mapreduce](http://hbase.apache.org/book.html#mapreduce)

## Hbase with Hadoop

```
public static class MyTableReducer extends
        TableReducer<Text, IntWritable, ImmutableBytesWritable>  {

    public void reduce(Text key,
                Iterable<IntWritable> values, Context context) ….. {
        int i = 0;
        for (IntWritable val : values) {
            i += val.get();
        }
        Put put = new Put(Bytes.toBytes(key.toString()));
        put.add(Bytes.toBytes("cf"),
                Bytes.toBytes("count"), Bytes.toBytes(i));
        context.write(null, put);
    }
}
```

More info : http://hbase.apache.org/book.html#mapreduce

## Hands-on HBase MapReduce Programming*

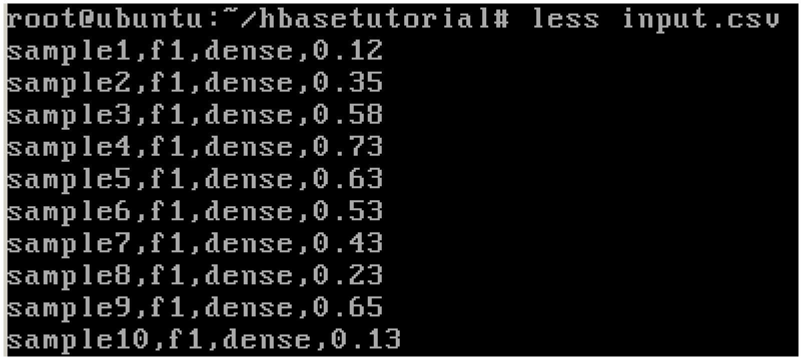
- HBase MapReduce API

```
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.mapreduce.TableMapper;
import org.apache.hadoop.hbase.mapreduce.TableReducer;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
import org.apache.hadoop.hbase.util.Bytes;
```

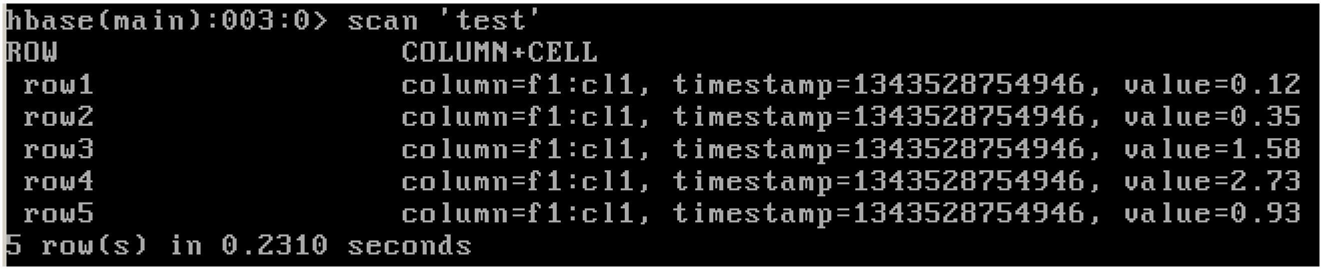
* Developed by Li Hui

# Hands-on: load CSV file into HBase table with MapReduce

- CSV represent for comma separate values
- CSV file is common file in many scientific fields



```
root@ubuntu:~/hbasetutorial# less input.csv
sample1,f1,dense,0.12
sample2,f1,dense,0.35
sample3,f1,dense,0.58
sample4,f1,dense,0.73
sample5,f1,dense,0.63
sample6,f1,dense,0.53
sample7,f1,dense,0.43
sample8,f1,dense,0.23
sample9,f1,dense,0.65
sample10,f1,dense,0.13
```



```
hbase(main):003:0> scan 'test'
ROW                    COLUMN+CELL
 row1                  column=f1:cl1, timestamp=1343528754946, value=0.12
 row2                  column=f1:cl1, timestamp=1343528754946, value=0.35
 row3                  column=f1:cl1, timestamp=1343528754946, value=1.58
 row4                  column=f1:cl1, timestamp=1343528754946, value=2.73
 row5                  column=f1:cl1, timestamp=1343528754946, value=0.93
5 row(s) in 0.2310 seconds
```

# Hands-on: load CSV file into HBase table with MapReduce

- Main entry point of program

```
public static void main(String[] args) throws Exception {
		Configuration conf = HBaseConfiguration.create();
		String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
		if(otherArgs.length != 2) {
		System.err.println("Wrong number of arguments: " + otherArgs.length);
		System.err.println("Usage:  <csv file>  <hbase table name>");
		System.exit(-1);
		}//end if
		Job job = configureJob(conf, otherArgs);
		System.exit(job.waitForCompletion(true) ? 0 : 1);
}//main
```

INDIANA UNIVERSITY

# Hands-on: load CSV file into HBase table with MapReduce

- Configure HBase MapReduce job

```
public static Job configureJob(Configuration conf, String [] args)  throws IOException {
        Path inputPath = new Path(args[0]);
        String tableName = args[1];
        Job job = new Job(conf,  tableName);
        job.setJarByClass(CSV2HBase.class);
        FileInputFormat.setInputPaths(job, inputPath);
        job.setInputFormatClass(TextInputFormat.class);
        job.setMapperClass(CSV2HBase.class);
        TableMapReduceUtil.initTableReducerJob(tableName, null, job);
        job.setNumReduceTasks(0);
        return job;
}//public static Job configure
```

# Hands-on: load CSV file into HBase table with MapReduce

- The map function

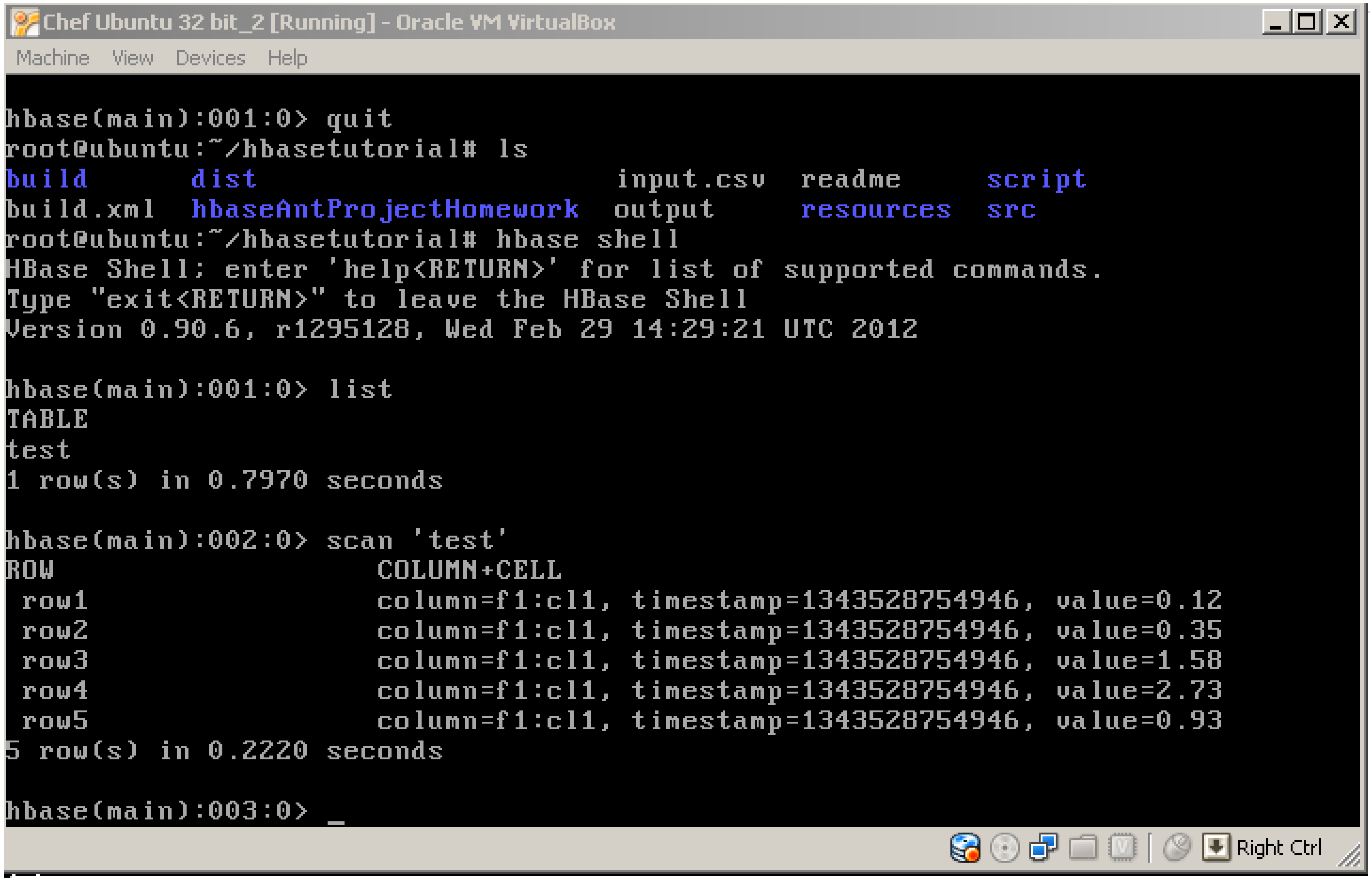
```
public void map(LongWritable key, Text line, Context context)    throws IOException {
   // Input is a CSV file Each map() is a single line, where the key is the line number
   // Each line is comma-delimited; row,family,qualifier,value
   String [] values = line.toString().split(",");
   if(values.length != 4) {      return;      }
   byte [] row = Bytes.toBytes(values[0]);
   byte [] family = Bytes.toBytes(values[1]);
   byte [] qualifier = Bytes.toBytes(values[2]);
   byte [] value = Bytes.toBytes(values[3]);
   Put put = new Put(row);
   put.add(family, qualifier, value);
   try {
     context.write(new ImmutableBytesWritable(row), put);
   } catch (InterruptedException e) {      e.printStackTrace();      }
  if(++count % checkpoint == 0) {
     context.setStatus("Emitting Put " + count);
   }   } }
```

# Hands-on: steps to load CSV file into HBase table with MapReduce

1. Check Hbase installation in Ubuntu Sandbox
   1. http://salsahpc.indiana.edu/ScienceCloud/virtualbox_appliance_guide.html
   2. echo $HBASE_HOME
2. Start Hadoop and Hbase cluster
   1. cd $HADOOP_HOME
   2. . MultiNodesOneClickStartUp.sh $JAVA_HOME nodes
   3. start-hbase.sh
3. Create hbase table with specified data schema
   1. hbase shell
   2. create “csv2hbase”,”f1”
   3. quit
4. Compile the program with Ant
   1. cd ~/hbasetutorial
   2. ant
5. Upload input.csv into HDFS
   1. hadoop dfs –mkdir input
   2. hadoop dfs –copyFromLocal input.csv input/input.csv
6. Run the program:
   hadoop  jar dist/lib/cglHBaseSummerSchool.jar iu.pti.hbaseapp.CSV2HBase input/input.csv “csv2hbase”
7. Check inserted records in Hbase table
   1. hbase shell
   2. scan “csv2hbase”

INDIANA UNIVERSITY

# Hands-on: load CSV file into HBase table with MapReduce


Chef Ubuntu 32 bit_2 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
hbase(main):001:0> quit
root@ubuntu:~/hbasetutorial# ls
build      dist                      input.csv  readme     script
build.xml  hbaseAntProjectHomework   output     resources  src
root@ubuntu:~/hbasetutorial# hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.90.6, r1295128, Wed Feb 29 14:29:21 UTC 2012

hbase(main):001:0> list
TABLE
test
1 row(s) in 0.7970 seconds

hbase(main):002:0> scan 'test'
ROW                    COLUMN+CELL
 row1                  column=f1:cl1, timestamp=1343528754946, value=0.12
 row2                  column=f1:cl1, timestamp=1343528754946, value=0.35
 row3                  column=f1:cl1, timestamp=1343528754946, value=1.58
 row4                  column=f1:cl1, timestamp=1343528754946, value=2.73
 row5                  column=f1:cl1, timestamp=1343528754946, value=0.93
5 row(s) in 0.2220 seconds

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

- HBase builtin “importtsv” commands supports importing CSV

# Other Features

- Bulk loading
  - HFileOutputFormat
- Multiple masters
- In memory column families
- Block cache
- Bloom filters

## Coming up…

- Introduction to Pig
- Demo
  - Search Engine System with MapReduce Technologies (Hadoop/HDFS/**HBase**/Pig)

## Questions ☺