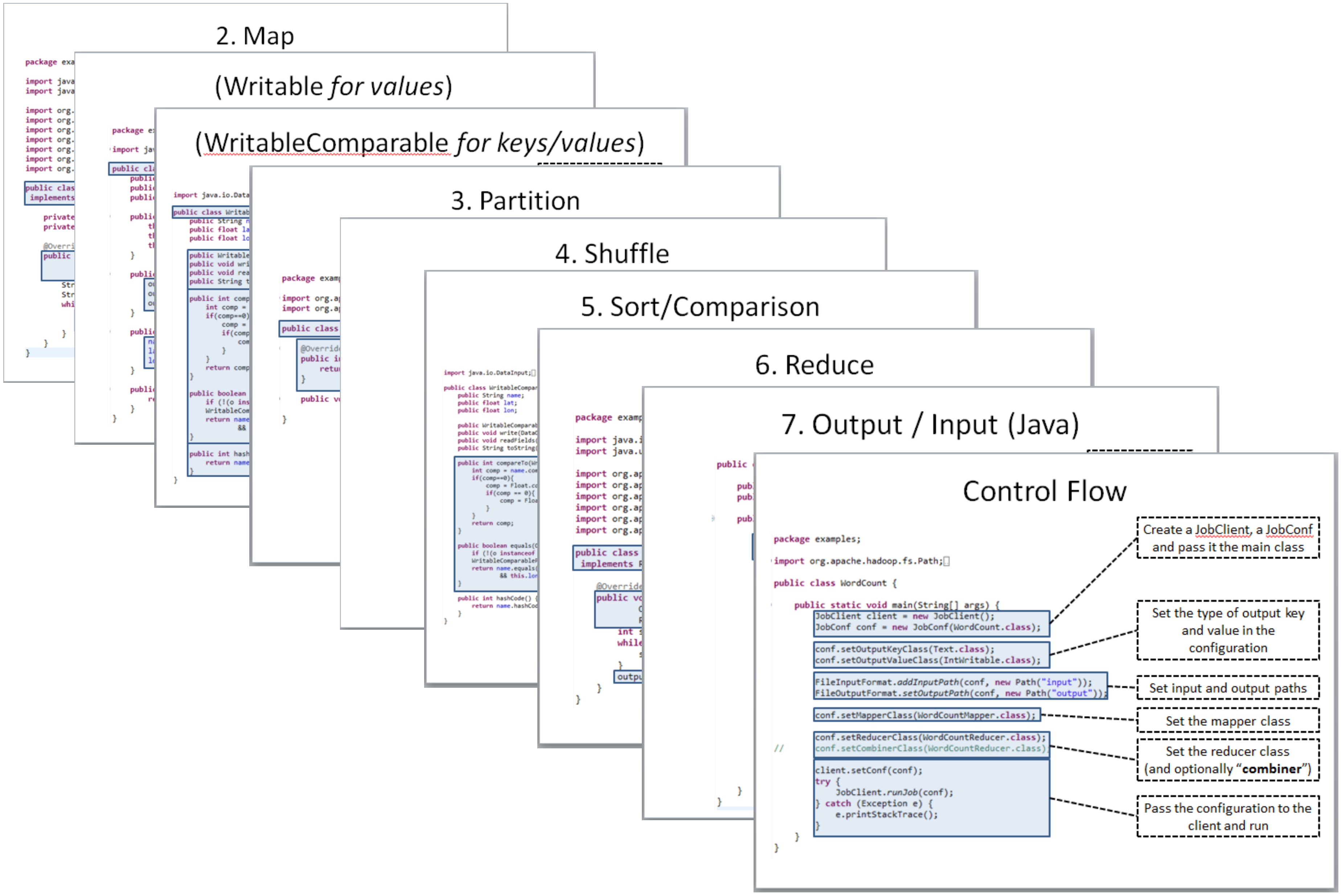
CC5212-1
PROCESAMIENTO MASIVO DE DATOS
OTOÑO 2016

Lecture 6: DFS & MapReduce III

Aidan Hogan
aidhog@gmail.com

Apache Hadoop (Java)



2. Map

(Writable *for values*)

(WritableComparable *for keys/values*)

3. Partition

4. Shuffle

5. Sort/Comparison

6. Reduce

7. Output / Input (Java)

Control Flow

```
package examples;

import org.apache.hadoop.fs.Path;

public class WordCount {

    public static void main(String[] args) {
        JobClient client = new JobClient();
        JobConf conf = new JobConf(WordCount.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(conf, new Path("input"));
        FileOutputFormat.setOutputPath(conf, new Path("output"));

        conf.setMapperClass(WordCountMapper.class);

        conf.setReducerClass(WordCountReducer.class);
//      conf.setCombinerClass(WordCountReducer.class);

        client.setConf(conf);
        try {
            JobClient.runJob(conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Create a JobClient, a JobConf and pass it the main class

Set the type of output key and value in the configuration

Set input and output paths

Set the mapper class

Set the reducer class (and optionally "**combiner**")

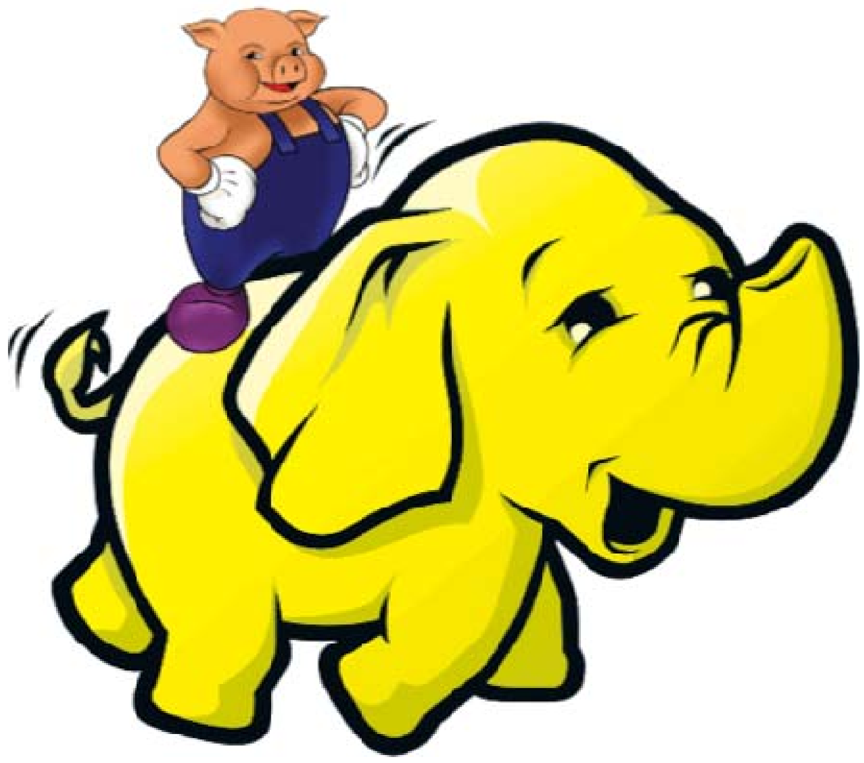
Pass the configuration to the client and run

An Easier Way?

APACHE PIG: OVERVIEW

Apache Pig

- Create MapReduce programs to **run on Hadoop**
- Use a high-level “scripting” language called **Pig Latin**
- Can embed **User Defined Functions**: call a Java function (or Python, Ruby, etc.)
- Based on **Pig Relations**



Pig Latin: Hello Word Count

```
input_lines = LOAD '/tmp/book.txt' AS (line:chararray);
```

```
-- Extract words from each line and put them into a pig bag
-- datatype, then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- filter out any words that are just white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';

-- create a group for each word
word_groups = GROUP filtered_words BY word;
```

Map

Reduce

```
-- count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;
```

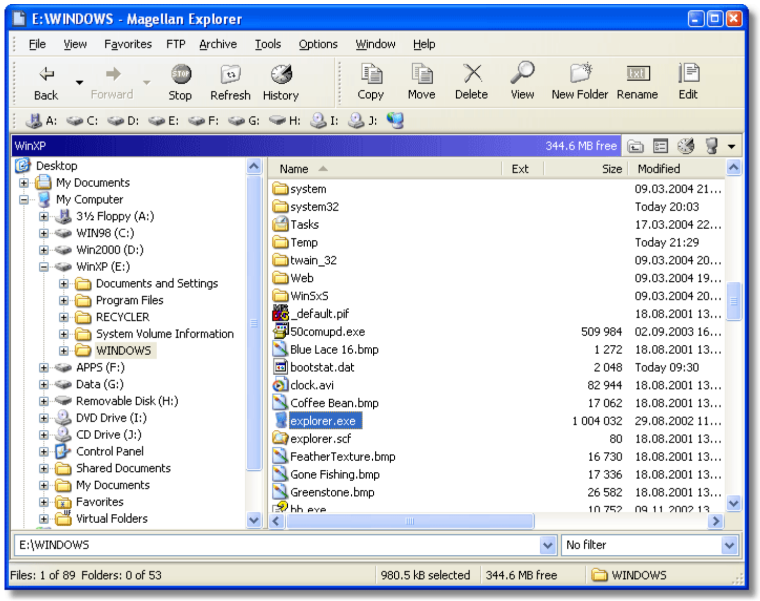
```
-- order the records by count
ordered_word_count = ORDER word_count BY count DESC;
```

Map + Reduce

```
STORE ordered_word_count INTO '/tmp/book-word-count.txt';
```

Any guesses which line(s) correpond to Map, Reduce?

Pig: Local Mode vs. MapReduce Mode



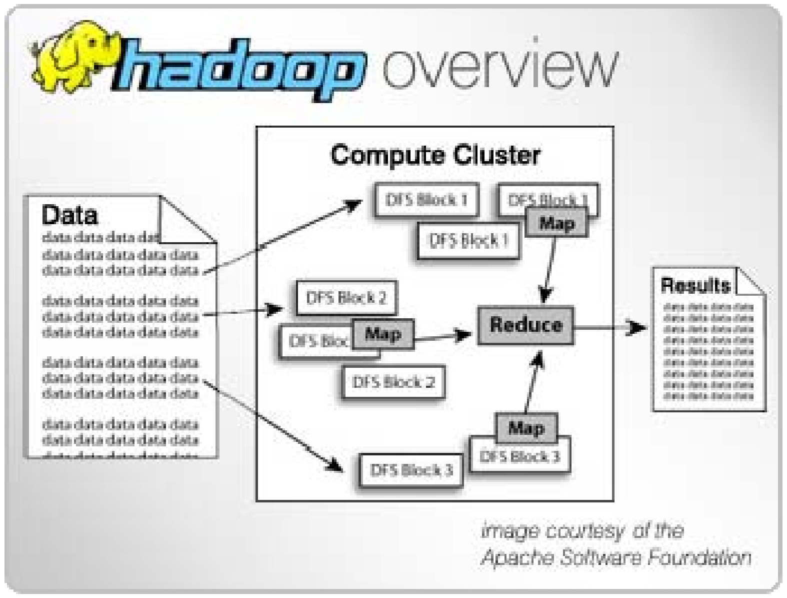
WinXP 344.6 MB free

Desktop
- My Documents
- My Computer
  - 3½ Floppy (A:)
  - WIN98 (C:)
  - Win2000 (D:)
  - WinXP (E:)
    - Documents and Settings
    - Program Files
    - RECYCLER
    - System Volume Information
    - WINDOWS
  - APPS (F:)
  - Data (G:)
  - Removable Disk (H:)
  - DVD Drive (I:)
  - CD Drive (J:)
  - Control Panel
  - Shared Documents
  - My Documents
- Favorites
- Virtual Folders

Name	Ext	Size	Modified
system			09.03.2004 21...
system32			Today 20:03
Tasks			17.03.2004 22...
Temp			Today 21:29
twain_32			09.03.2004 20...
Web			09.03.2004 19...
WinSxS			09.03.2004 20...
_default.pif			18.08.2001 13...
50comupd.exe		509 984	02.09.2003 16...
Blue Lace 16.bmp		1 272	18.08.2001 13...
bootstat.dat		2 048	Today 09:30
clock.avi		82 944	18.08.2001 13...
Coffee Bean.bmp		17 062	18.08.2001 13...
explorer.exe		1 004 032	29.08.2002 11...
explorer.scf		80	18.08.2001 13...
FeatherTexture.bmp		16 730	18.08.2001 13...
Gone Fishing.bmp		17 336	18.08.2001 13...
Greenstone.bmp		26 582	18.08.2001 13...
hh.exe		10 752	09.11.2002 13...

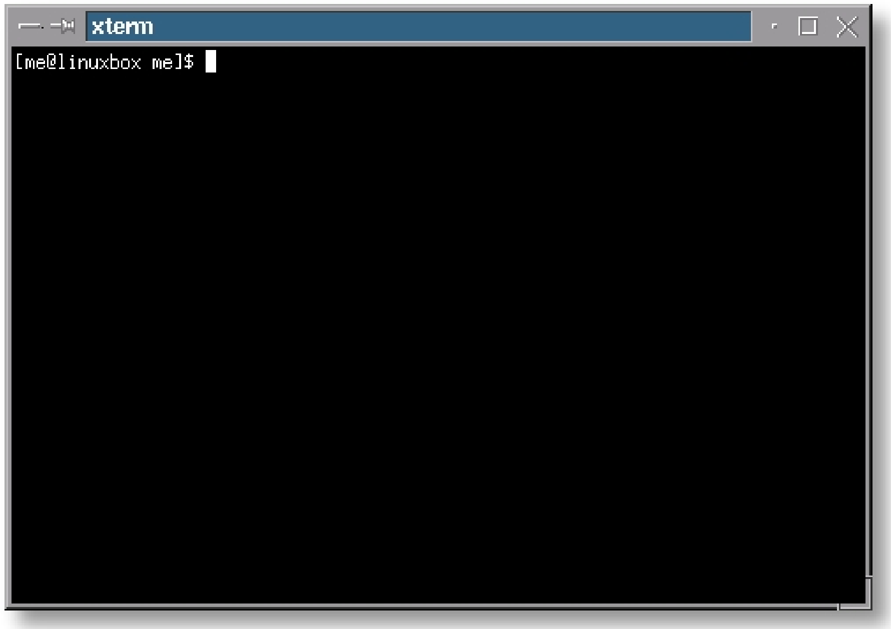
E:\WINDOWS No filter

Files: 1 of 89 Folders: 0 of 53 980.5 kB selected 344.6 MB free WINDOWS



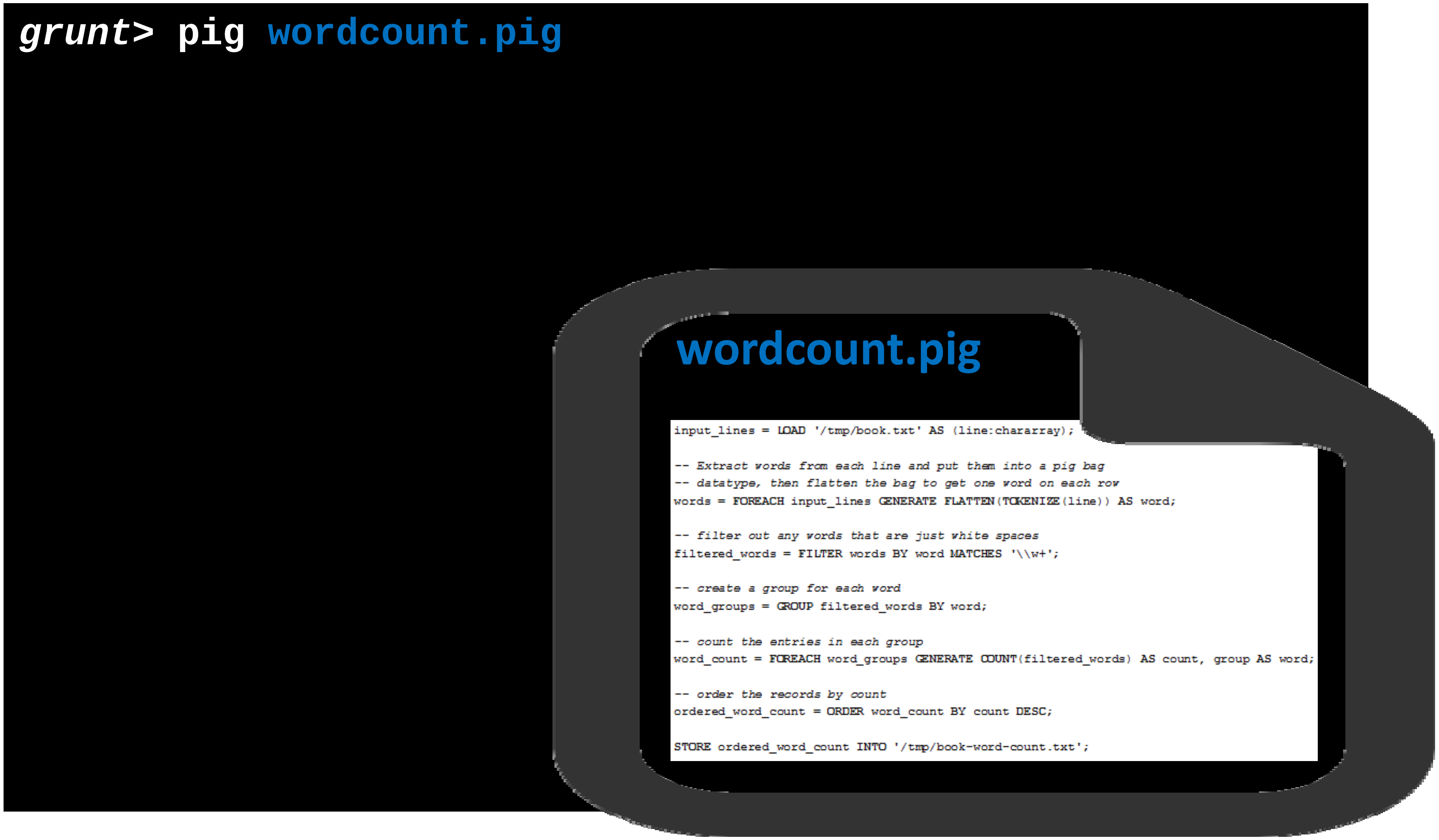
Three Ways to Execute Pig: (i) Grunt

```
grunt> in_lines = LOAD '/tmp/book.txt' AS (line:chararray);
grunt> words = FOREACH in_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
grunt> filtered_words = FILTER words BY word MATCHES '\\w+';
grunt> …
…
grunt> STORE ordered_word_count INTO '/tmp/book-word-count.txt';
```



```
[me@linuxbox me]$
```

Three Ways to Execute Pig: (ii) Script



```
grunt> pig wordcount.pig
```

wordcount.pig

```
input_lines = LOAD '/tmp/book.txt' AS (line:chararray);

-- Extract words from each line and put them into a pig bag
-- datatype, then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- filter out any words that are just white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';

-- create a group for each word
word_groups = GROUP filtered_words BY word;

-- count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS word;

-- order the records by count
ordered_word_count = ORDER word_count BY count DESC;

STORE ordered_word_count INTO '/tmp/book-word-count.txt';
```

Three Ways to Execute Pig: (iii) Embedded

```java
package scratch;

import org.apache.pig.PigServer;

public class PigLatinWordCount {

    public static void main(String[] args) {
        String inputFile = args[0];
        String outputFile = args[1];
        try {
            PigServer pigServer = new PigServer("local");
            pigServer.registerQuery("in_lines = LOAD '" + inputFile + "' AS (line:chararray);");
            pigServer.registerQuery("words = FOREACH in_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;");
            // ...
            // ...
            pigServer.store("ordered_word_count", outputFile);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

}
```

APACHE PIG: LIDER EXAMPLE

Pig: Products by Hour

transact.txt

```
customer412    1L_Leche          2014-03-31T08:47:57Z      $900
customer412    Nescafe           2014-03-31T08:47:57Z    $2.000
customer412    Nescafe           2014-03-31T08:47:57Z    $2.000
customer413    400g_Zanahoria    2014-03-31T08:48:03Z    $1.240
customer413    El_Mercurio       2014-03-31T08:48:03Z      $500
customer413    Gillette_Mach3    2014-03-31T08:48:03Z    $8.250
customer413    Santo_Domingo     2014-03-31T08:48:03Z    $2.450
customer413    Nescafe           2014-03-31T08:48:03Z    $2.000
customer414    Rosas             2014-03-31T08:48:24Z    $7.000
customer414    Chocolates        2014-03-31T08:48:24Z    $9.230
customer414    300g_Frutillas    2014-03-31T08:48:24Z    $1.230
cu
cu
…
```

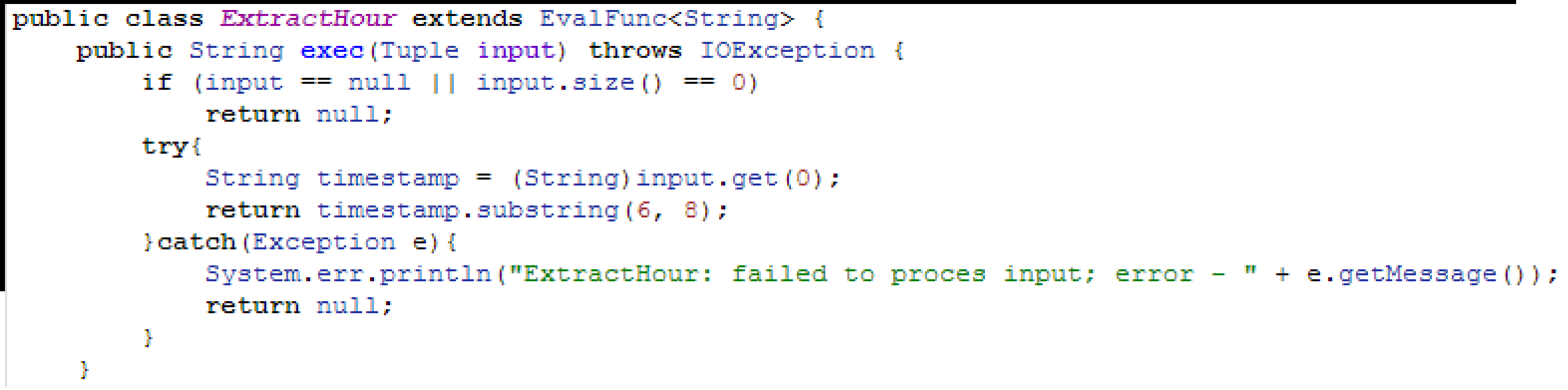
Your boss in Lider Headquarters tells you to find out the frequency of premium items (price>$1.000) sold per hour counting duplicate items from each customer once …

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
```

User-defined-functions written in Java (or Python, Ruby, etc. …)

userDefinedFunctions.jar



```java
public class ExtractHour extends EvalFunc<String> {
    public String exec(Tuple input) throws IOException {
        if (input == null || input.size() == 0)
            return null;
        try{
            String timestamp = (String)input.get(0);
            return timestamp.substring(6, 8);
        }catch(Exception e){
            System.err.println("ExtractHour: failed to proces input; error - " + e.getMessage());
            return null;
        }
    }
```

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
```

View data as a (streaming) relation with fields (cust, item, etc.) and tuples (data rows) …

raw:

cust	item	time	price
customer412	1L_Leche	2014-03-31T08:47:57Z	$900
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer413	400g_Zanahoria	2014-03-31T08:48:03Z	$1.240
…	…	…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
```

Filter tuples depending on their value for a given attribute (in this case, price < 1000)

premium:

cust	item	time	price
customer412	1L_Leche	2014-03-31T08:47:57Z	$900
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer413	400g_Zanahoria	2014-03-31T08:48:03Z	$1.240
…	…	…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD 'transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
```

hourly:

cust	item	hour	price
customer412	Nescafe	08	$2.000
customer412	Nescafe	08	$2.000
customer413	400g_Zanahoria	08	$1.240
customer413	Gillette_Mach3	08	$8.250
…	…	…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
```

unique:

cust	item	hour	price
customer412	Nescafe	08	$2.000
customer412	Nescafe	08	$2.000
customer413	400g_Zanahoria	08	$1.240
customer413	Gillette_Mach3	08	$8.250
…	…	…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
grunt> hrItem = GROUP unique BY (item, hour);
```

unique:

cust	item	hour	price
customer412	Nescafe	08	$2.000
customer413	400g_Zanahoria	08	$1.240
customer413	Gillette_Mach3	08	$8.250
customer413	Santo_Domingo	08	$2.450
…	…	…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
grunt> hrItem = GROUP unique BY (item, hour);
```

hrItem:

[item,hour]	cust	item	hour	price
[Nescafe,08]	customer412	Nescafe	08	$2.000
	customer413	Nescafe	08	$2.000
	customer415	Nescafe	08	$2.000
[400g_Zanahoria,08]	customer413	400g_Zanahoria	08	$1.240
…	…	…	…	…

Pig: Products by Hour

```
grunt>  REGISTER userDefinedFunctions.jar
grunt>  raw = LOAD 'transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt>  premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt>  hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt>  unique = DISTINCT hourly;
grunt>  hrItem = GROUP unique BY (item, hour);
grunt>  hrItemCnt = FOREACH hrItem GENERATE flatten($0), COUNT($1) AS count;
```

hrItem:

[item,hour]	cust	item	hour	price
[Nescafe,08]	customer412	Nescafe	08	$2.000
	customer413	Nescafe	08	$2.000
count	customer415	Nescafe	08	$2.000
[400g_Zanahoria,08]	customer413	400g_Zanahoria	08	$1.240
…	…	…	…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD 'transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
grunt> hrItem = GROUP unique BY (item, hour);
grunt> hrItemCnt = FOREACH hrItem GENERATE flatten($0), COUNT($1) AS count;
```

hrItemCnt:

[item,hour]	count
[400g_Zanahoria,08]	1
[Nescafe,08]	3
…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
grunt> hrItem = GROUP unique BY (item, hour);
grunt> hrItemCnt = FOREACH hrItem GENERATE flatten($0), COUNT($1) AS count;
grunt> hrItemCntSorted = ORDER hrItemCnt BY count DESC;
```

hrItemCnt:

[item,hour]	count
[400g_Zanahoria,08]	1
[Nescafe,08]	3
…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
grunt> hrItem = GROUP unique BY (item, hour);
grunt> hrItemCnt = FOREACH hrItem GENERATE flatten($0), COUNT($1) AS count;
grunt> hrItemCntSorted = ORDER hrItemCnt BY count DESC;
```

hrItemCntSorted:

[item,hour]	count
[Nescafe,08]	3
[400g_Zanahoria,08]	1
…	…

Pig: Products by Hour

```
grunt> REGISTER userDefinedFunctions.jar
grunt> raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
grunt> premium = FILTER raw BY org.udf.MinPrice1000(price);
grunt> hourly = FOREACH premium GENERATE cust, item,
org.udf.ExtractHour(time) AS hour, price;
grunt> unique = DISTINCT hourly;
grunt> hrItem = GROUP unique BY (item, hour);
grunt> hrItemCnt = FOREACH hrItem GENERATE flatten($0), COUNT($1) AS count;
grunt> hrItemCntSorted = ORDER hrItemCnt BY count DESC;
grunt> STORE hrItemCntSorted INTO ‘output.txt’
```

hrItemCntSorted:

[item,hour]	count
[Nescafe,08]	3
[400g_Zanahoria,08]	1
…	…

APACHE PIG: SCHEMA

Pig Relations

- Pig Relations: Like relational tables
  - Except tuples can be "jagged"
  - Fields in the same column don't need to be same type
  - Relations are by default unordered
- Pig Schema: Names for fields, etc.

```
… AS (cust, item, time, price);
```

cust	item	time	price
customer412	1L_Leche	2014-03-31T08:47:57Z	$900
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer413	400g_Zanahoria	2014-03-31T08:48:03Z	$1.240
…	…	…	…

Pig Fields

- Pig Fields:
  - Reference using name
    - `premium = FILTER raw BY org.udf.MinPrice1000(price);`
  - … or position
    - `premium = FILTER raw BY org.udf.MinPrice1000($3);`

More readable!

Starts at zero.

cust	item	time	price
customer412	1L_Leche	2014-03-31T08:47:57Z	$900
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer412	Nescafe	2014-03-31T08:47:57Z	$2.000
customer413	400g_Zanahoria	2014-03-31T08:48:03Z	$1.240
…	…	…	…

Pig Simple Types

- Pig Types:

```
LOAD ‘transact.txt' USING PigStorage('\t') AS
(cust:charArray, item:charArray, time:datetime,
price:int);
```

- int, long, float, double, biginteger, bigdecimal, boolean, chararray (string), bytearray (blob), datetime

Pig Types: Duck Typing

- What happens if you omit types?
  - Fields default to bytearray
  - Implicit conversions if needed (~duck typing)

```
A = LOAD 'data' AS (cust, item, hour, price);
B = FOREACH A GENERATE hour + 4 % 24;
C = FOREACH A GENERATE hour + 4f % 24;
```

hour an integer

hour a float

Pig Complex Types: Tuple

```
cat data;
(3,8,9) (4,5,6)
(1,4,7) (3,7,5)
(2,5,8) (9,5,8)

A = LOAD 'data' AS (t1:tuple(t1a:int,t1b:int,t1c:int),t2:tuple(t2a:int,t2b:int,t2c:int));

DUMP A;
((3,8,9),(4,5,6)) ((1,4,7),(3,7,5)) ((2,5,8),(9,5,8))

X = FOREACH A GENERATE t1.t1a,t2.$0;
```

A:

t1			t2		
t1a	t1b	t1c	t2a	t2b	t2c
3	8	9	4	5	6
1	4	7	3	7	5
2	5	8	9	5	8

Pig Complex Types: Tuple

```
cat data;
(3,8,9) (4,5,6)
(1,4,7) (3,7,5)
(2,5,8) (9,5,8)

A = LOAD 'data' AS (t1:tuple(t1a:int,t1b:int,t1c:int),t2:tuple(t2a:int,t2b:int,t2c:int));

DUMP A;
((3,8,9),(4,5,6)) ((1,4,7),(3,7,5)) ((2,5,8),(9,5,8))

X = FOREACH A GENERATE t1.t1a,t2.$0;
DUMP X;
(3,4) (1,3) (2,9)
```

X:

$0	$1
3	4
1	3
2	9

Pig Complex Types: Bag

```
cat data;
(3,8,9)
(2,3,6)
(1,4,7)
(2,5,8)

A = LOAD 'data' AS (c1:int, c2:int, c3:int);
B = GROUP A BY c1;
```

A:

c1	c2	c3
3	8	9
2	3	6
1	4	7
2	5	8

Pig Complex Types: Bag

```
cat data;
(3,8,9)
(2,3,6)
(1,4,7)
(2,5,8)

A = LOAD 'data' AS (c1:int, c2:int, c3:int);
B = GROUP A BY c1;
DUMP B;
(1,{(1,4,7)})
(2,{(2,5,8),(2,3,6)})
(3,{(3,8,9)})
```

B:

group (c1)	A		
	c1	c2	c3
3	3	8	9
2	2	3	6
	2	5	8
1	1	4	7

Pig Complex Types: Map

```
cat prices;
[Nescafe#”$2.000”]
[Gillette_Mach3#”$8.250”]

A = LOAD ‘prices’ AS (M:map []);
```

Pig Complex Types: Summary

- tuple: A row in a table / a list of fields
  - e.g., (customer412, Nescafe, 08, $2.000)
- bag: A set of tuples (allows duplicates)
  - e.g., { (cust412, Nescafe, 08, $2.000), (cust413, Gillette_Mach3, 08, $8.250) }
- map: A set of key–value pairs
  - e.g., [Nescafe#$2.000]

APACHE PIG: OPERATORS

Pig Atomic Operators

- Comparison
  ==, !=, >, <, >=, <=, `matches` (regex)
- Arithmetic
  + , −, *, /
- Reference
  tuple.field, map#value
- Boolean
  AND, OR, NOT
- Casting

Pig Conditionals

- Ternary operator:

```
hr12 = FOREACH item GENERATE hour%12, (hour>12 ? ’pm’ : ’am’);
```

- Cases:

```
X = FOREACH A GENERATE hour%12, (
      CASE
            WHEN hour>12 THEN ‘pm’
            ELSE ‘am’
      END
);
```

Pig Aggregate Operators

Can GROUP multiple items or COGROUP single item (COGROUP considered more readable for multiple items)

- Grouping:
  - GROUP: group on a single relation
    - **GROUP** premium **BY** (item, hour);
  - COGROUP: group multiple relations
    - **COGROUP** premium **BY** (item, hour), cheap **BY** (item, hour);
- Aggregate Operations:
  - AVG, MIN, MAX, SUM, COUNT, SIZE, CONCAT

Pig Joins

```
cat data1;
(Nescafe,08,120)
(El_Mercurio,08,142)
(Nescafe,09,153)

cat data2;
(2000,Nescafe)
(8250, Gillette_Mach3)
(500, El_Mercurio)

A = LOAD 'data1' AS (prod:charArray, hour:int, count:int);
B = LOAD 'data2' AS (price:int, name:charArray);
X = JOIN A BY prod, B BY name;

DUMP X:
(El_Mercurio,08,142, 500, El_Mercurio)
(Nescafe,08,120, 2000,Nescafe)
(Nescafe,09,153, 2000,Nescafe)
```

X:

prod	hour	count	price	name
Nescafe	08	120	2000	Nescafe
Nescafe	09	153	2000	Nescafe
El_Mercurio	08	142	500	El_Mercurio

Pig Joins

- Inner join: As shown (default)
- Self join: Copy an alias and join with that
- Outer joins:
  - LEFT / RIGHT / FULL
- Cross product:
  - CROSS

You guys know (or remember ☺) what an INNER JOIN is versus an OUTER JOIN / LEFT / RIGHT / FULL versus a CROSS PRODUCT?

Pig Aggregate/Join Implementations

- Custom partitioning / number of reducers:
  - PARTITION BY specifies a UDF for partitioning
  - PARALLEL specifies number of reducers

```
X = JOIN A BY prod, B BY name PARTITION BY org.udp.Partitioner
PARALLEL 5;
```

```
X = GROUP A BY hour PARTITION BY org.udp.Partitioner PARALLEL 5;
```

Pig: Disambiguate

```
cat data1;
(Nescafe,08,120)
(El_Mercurio,08,142)
(Nescafe,09,153)

cat data2;
(2000,Nescafe)
(8250,Gillette_Mach3)
(500,El_Mercurio)

A = LOAD 'data1' AS (prodName:charArray, hour:int, count:int);
B = LOAD 'data2' AS (price:int, prodName:charArray);
X = JOIN A BY prodName, B BY prodName;

DUMP X:
(El_Mercurio,08,142,500,El_Mercurio)
(Nescafe,08,120, 2000,Nescafe)
(Nescafe,09,153, 2000,Nescafe)

Y = FOREACH X GENERATE prodName
Y = FOREACH X GENERATE A::prodName
```

which prodName?

Pig: Split

```
raw = LOAD ‘transact.txt' USING PigStorage('\t') AS (cust, item, time, price);
numeric = FOREACH raw GENERATE cust item time org.udf.RemoveDollarSign(price) AS price;
SPLIT numeric INTO cheap IF price<1000, premium IF price>=1000;
```

numeric:

cust	item	time	price
customer412	1L_Leche	2014-03-31T08:47:57Z	900
customer412	Nescafe	2014-03-31T08:47:57Z	2.000
customer412	Nescafe	2014-03-31T08:47:57Z	2.000
customer413	400g_Zanahoria	2014-03-31T08:48:03Z	1.240
…	…	…	…

cheap:

cust	item	time	price
customer412	1L_Leche	2014-03-31T08:47:57Z	900
…	…	…	…

premium:

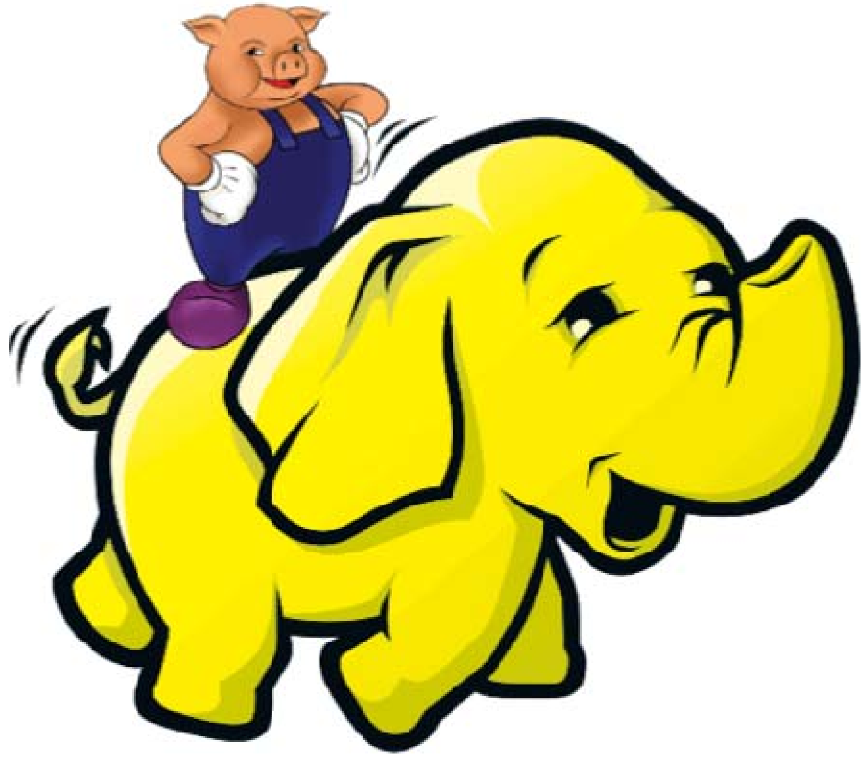
cust	item	time	price
customer412	Nescafe	2014-03-31T08:47:57Z	2.000
customer412	Nescafe	2014-03-31T08:47:57Z	2.000
customer413	400g_Zanahoria	2014-03-31T08:48:03Z	1.240
…	…	…	…

Pig: Other Operators

- FILTER: Filter tuples by an expression
- LIMIT: Only return a certain number of tuples
- MAPREDUCE: Run a native Hadoop .jar
- ORDER BY: Sort tuples
- SAMPLE: Sample tuples
- UNION: Concatenate two relations

Pig translated to MapReduce in Hadoop

- Pig is only an interface/scripting language for MapReduce



JUST TO MENTION …

Apache Hive

- SQL-style language that compiles into MapReduce jobs in Hadoop

- Similar to Apache Pig but …
  - Pig more procedural whilst Hive more declarative

RECAP …

Programming

Apache Pig (Latin)

- Allows for scripting MapReduce jobs:
- Procedural, but makes use of relational algebra
- Three ways to run:
  1. Interactive command line
  2. Batch script
  3. Call from Java

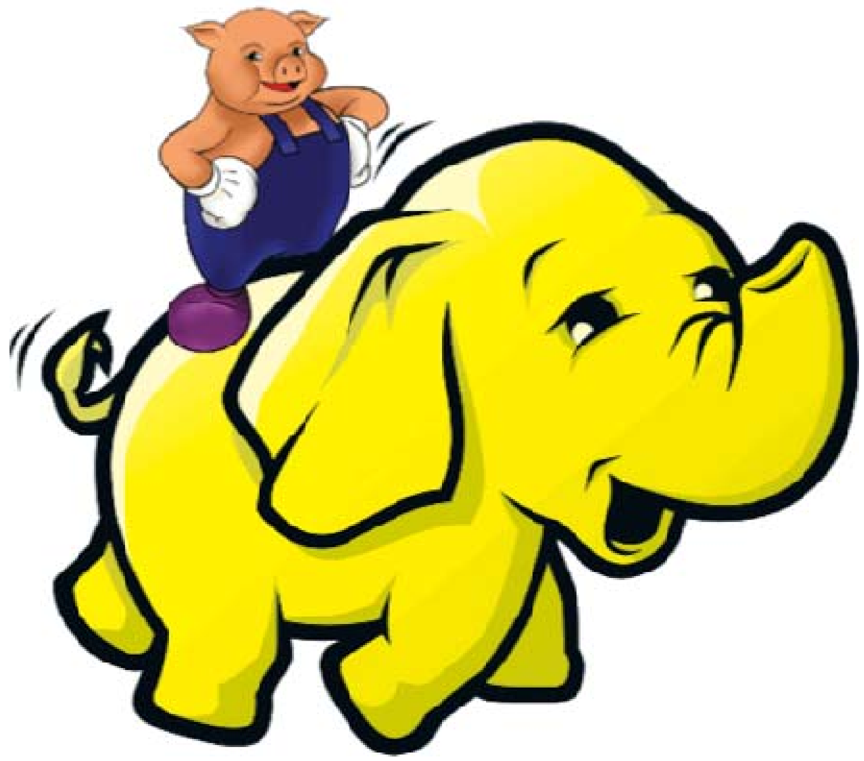
Apache Pig (Latin)

- Schema based on relations:
  - A bit like databases

- Some basic programming operators:
  - arithmetic, boolean, conditions, casting
- Some relational algebra:
  - joins, groups, count, avg, sum, filter, limit, order by, etc.
- For everything else, there's user-defined functions

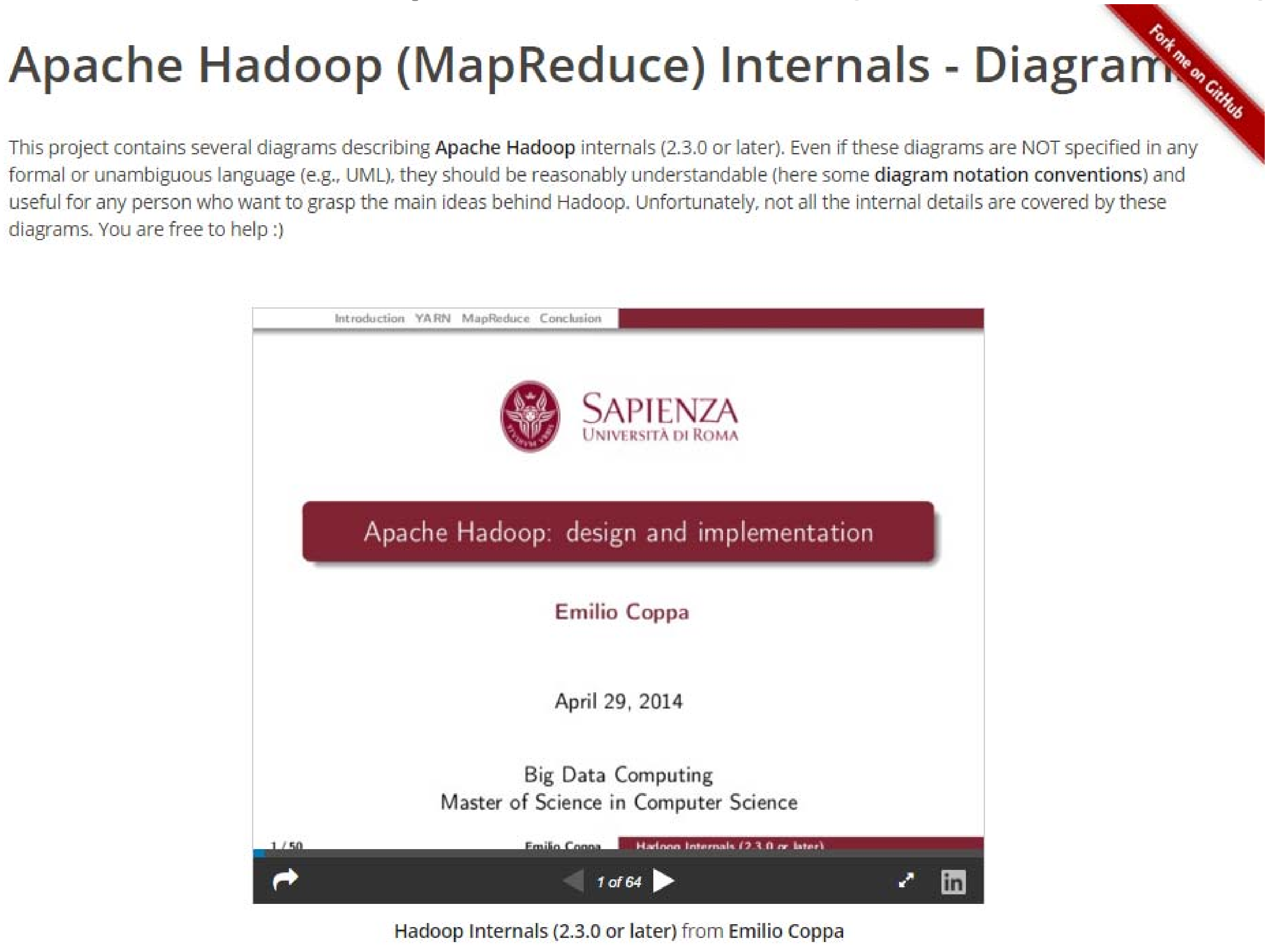
More reading

https://pig.apache.org/docs/r0.7.0/piglatin_ref2.html



CONCLUDING MAPREDUCE (FOR NOW) …

Apache Hadoop … Internals (if interested)



Apache Hadoop (MapReduce) Internals - Diagrams

This project contains several diagrams describing **Apache Hadoop** internals (2.3.0 or later). Even if these diagrams are NOT specified in any formal or unambiguous language (e.g., UML), they should be reasonably understandable (here some **diagram notation conventions**) and useful for any person who want to grasp the main ideas behind Hadoop. Unfortunately, not all the internal details are covered by these diagrams. You are free to help :)

Apache Hadoop: design and implementation

Emilio Coppa

April 29, 2014

Big Data Computing
Master of Science in Computer Science

Hadoop Internals (2.3.0 or later) from **Emilio Coppa**

http://ercoppa.github.io/HadoopInternals/

Questions