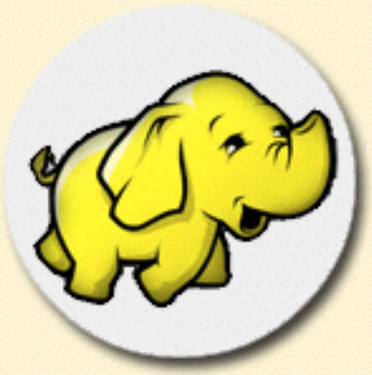


Welcome to

Big Data & Hadoop

Session

Session 5 - Pig & Pig Latin

+91-9538998962

sandeep@knowbigdata.com

WELCOME - KNOWBIGDATA

- Interact - Ask Questions
- Lifetime access of content
- Class Recording
- Cluster Access
- 24x7 support
- Real Life Project
- Quizzes & Certification Test
- 10 x (3hr class)
- Socio-Pro Visibility
- Mock Interviews

ABOUT ME

2014	**KnowBigData**	Founded
2014	**Amazon**	Built High Throughput Systems for Amazon.com site using in-house NoSql.
2012		
2012	**InMobi**	Built Recommender after churning 200 TB
2011		
	tBits Global	Founded tBits Global Built an enterprise grade Document Management System
2006		
	D.E.Shaw	Built the big data systems before the term was coined
2002		
2002	**IIT Roorkee**	Finished B.Tech somehow.

COURSE CONTENT

I	Understanding BigData, Hadoop Architecture
II	Environment Overview, MapReduce Basics
III	Adv MapReduce & Testing
IV	Pig & Pig Latin
V	Analytics using Hive
VI	NoSQL, HBASE
VII	Oozie, Mahout,
VIII	Zookeeper, Apache Storm
IX	Apache Flume, Apache Spark
X	YARN, Big Data Sets & Project Assignment

TODAY'S CLASS

- Introduction
- Wordcount example
- Pig Latin vs SQL
- Pig Latin vs MapReduce
- Use Cases
- Philosophy
- Installation
- Using from Cloud Labs
- Local Mode
- Command Line Args
- Prompt - Grunt
- Average Val Example
- Data Types - Scaler
- Data Types - Complex
- Schema
- Basic Contructions
- LOAD
- Diagnostic
- STORE & DUMP
- Foreach

TODAY'S CLASS

- FILTER
- Order by
- JOIN
- Limit & Sample
- Parallel
- Flatten
- Calling External Functions
- Foreach Nested
- Fragmented Joins
- Skewed Join - Merge
- Left Outer Join
- Right Outer Join
- Full Join
- CoGroup
- Union
- Cross
- Stream
- Params
- Splits
- Non-Linear Execution

ADV. MAP / REDUCE

Followup Questions?

PIG

An engine for executing data flows in parallel on Hadoop

- Language - Pig Latin
  - Provides operators: Join, order by, Filter …
- Runs on Hadoop. Makes use of both
  - HDFS
  - MapReduce

PIG - WORD COUNT EXAMPLE

```
-- Load input from the file named big.txt, and call the single
-- field in the record 'line'.
input = load ‘big.txt' as (line);

-- TOKENIZE splits the line into a field for each word.
-- flatten will take the collection of records returned by
-- TOKENIZE and produce a separate record for each one, calling the single
-- field in the record word.
words = foreach input generate flatten(TOKENIZE(line)) as word;

-- Now group them together by each word.
grpd  = group words by word;

-- Count them.
cntd  = foreach grpd generate group, COUNT(words);
-- Print out the results.
dump cntd;
```

PIG LATIN - A DATA FLOW LANGUAGE

Allows describing:

1. How data flows
2. Should be Read
3. Processed
4. Stored to multiple outputs in parallel

1. Complex Workflows
    1. Multiple Inputs are joined
    2. Data is split into multiple streams
2. No Ifs and fors

Map Reduce Programming vs Pig Latin

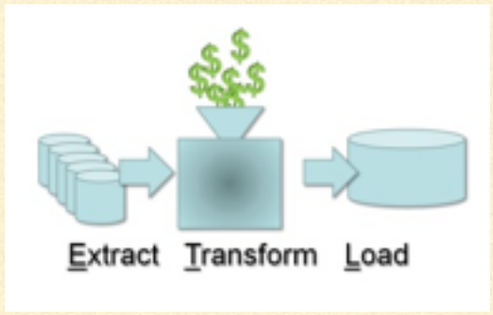
PIG LATIN	MAP REDUCE
Provides join, filter, group by, order by, union	Provides operations such as group by and order by indirectly. Joins are hard
Provides optimisation such are rebalancing the reducers	Same optimisations are really hard to write
Understand your scripts: Error checking & optimise	You code is opaque to the map reduce
Lower cost to write and maintain than Java code for MapReduce.	Requires more effort and time
Brief: 9lines	170 lines
Has limitations	More Control.

Pig Latin vs SQL

PIG LATIN	SQL
User describes exactly how to process the input data.	Allows users to describe what question they want answered
Long series of data operations	Around answering one question
Designed for Hadoop	Designed for RDBMS
Targets multiple data operations & parallel processing	For multiple data operations, the query gets complex
Can be used when schema is unknown, incomplete, or inconsistent,	requires well formatted

PIG - USE CASES

1. ETL
2. Research On Data
3. Iterative processing
    1. Batch processing



Map Reduce — PIG — SQL

PIG PHILOSOPHY

A. Pigs eat anything
   - Data: Relational, nested, or unstructured.
B. Pigs live anywhere
   - Parallel processing Language. Not only Hadoop
C. Pigs are domestic animals
   - Controllable, provides custom functions in java/python
   - Custom load and store methods
D. Pigs fly
   - Designed for performance

PIG - INSTALLING

1. Download 0.13.0 from Pig's release page
2. tar -xvfz pig-0.13.0.tar.gz
3. mv pig-0.13.0 /usr/local/pig
4. export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64
5. Run pig -x local and test if it is working
6. export PIG_CLASSPATH=/etc/hadoop
7. Add pig/bin folder to $PATH
8. Run pig and enjoy the grunts

PIG - USING FROM CLOUD LABS

- ssh student@hadoop1.knowbigdata.com (password is same as mentioned earlier). You can also try hadoop2, 3, 4

  pig

- In HDFS, you have the permissions to manipulate /users/student/

- The data has been uploaded to /users/student/data in HDFS

PIG - LOCAL MODE

- pig -x local
- Works without Hadoop installation
- Very useful for testing locally
- Improves the development pace

PIG - COMMAND LINE ARGUMENTS

- The first argument can be a pig script file: pig <myscript>
- To Execute a single command e.g.:
  - *pig -e ls*
- -version , for version
- -h see more options

PIG - GRUNT

- The prompt “grunt>”
- This is where all of the commands are typed
- You can also control hadoop from here
  - fs (-ls or -cat or any other hadoop command)
  - kill job
  - exec or run pig-script

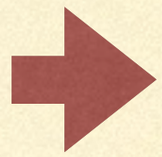
PIG - BASIC CONSTRUCTION

- Each processing step results in new set
- Case Insensitive
- Multiline comments: /* */
- Singleline comments: --

PIG - MY FIRST PIG SCRIPT - AVERAGE VAL

Local: data/NYSE_dividends
hdfs: /users/student/data/NYSE_dividends

NYSE	CPO	2009-12-30	0.11
NYSE	CPO	2009-09-28	0.12
NYSE	CPO	2009-06-26	0.13
NYSE	CCS	2009-10-28	0.41
NYSE	CCS	2009-04-29	0.43
NYSE	CIF	2009-12-09	.029
NYSE	CIF	2009-12-09	.028



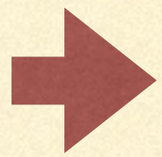
CPO	0.12
CCS	0.42
CIF	.0285

PIG - MY FIRST PIG SCRIPT - AVERAGE VAL

Local: data/NYSE_dividends

hdfs: /user/student/data/NYSE_dividends

NYSE	CPO	2009-12-30	0.11
NYSE	CPO	2009-09-28	0.12
NYSE	CPO	2009-06-26	0.13
NYSE	CCS	2009-10-28	0.41
NYSE	CCS	2009-04-29	0.43
NYSE	CIF	2009-12-09	.029
NYSE	CIF	2009-12-09	.028



CPO	0.12
CCS	0.42
CIF	.0285

```
1. divs = load '/user/student/data/NYSE_dividends' as (name, ticker, date, val);
2. dump divs
3. explain divs
4. grped = group divs by ticker;
5. avged = foreach grped generate group, AVG(divs.val);
6. store avged into '/user/student/data-out/avged';
7. fs -cat /user/student/data-out/avged/part*
```

PIG - DATA TYPES

Scaler

1. int
2. long
3. float - 4 bytes
4. double - 8 bytes
5. chararray
6. bytearray

PIG - DATA TYPES

Complex

1. Map
    1. ['name'#'bob', 'age'#55]
    2. chararray => another complex type or scaler
2. Tuple
    1. Fixed length, ordered collection
    2. Made up fields and their values.
    3. Think of it as a row in RDBMS and fields as columns
    4. Can refer the fields by position
    5. ('bob', 55, 12.3)
3. BAG
    1. Unordered collection of tuples
    2. {('ram', 55, 12.3), ('sally', 52, 11.2)}
        1. Two tuples each with three fields
    3. Analogous to SET of rows / dataset
    4. Does not need to fit in memory. Spills over to disk

PIG - DATA TYPES

Schemas

1. Not very particular about schema
2. You can either tell upfront, e.g.
    1. dividends = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray, date:chararray, dividend:float);
3. Or it will make best type guesses based on data
4. Schema declaration for complex types:
    1. a:map[] or a:map[int] …
    2. a:tuple or a:tuple(x, y) or a:tuple(x:int, y:bytearray,…)….
    3. a:bag{} or a:bag{t:(x:int, y:int)} or a:bag{t:(x, y)}
5. You can cast the data from one type to another if possible
    1. similar to Java by affixing "(type)"

MAP / REDUCE

BREAK for 10 mins

PIG - LOAD

- **divs = load 'NYSE_dividends'**
  - Loads the values guessing the datatype
  - Tab is used as separator
  - Absolute or Relative URL would work
- **divs = load 'NYSE_dividends' using PigStorage(',');**
  - To load comma separated values
- **divs = load 'NYSE_div*'**
  - You can specify the wildcard character such as '*'
- **divs = load 'NYSE_dividends' using HBaseStorage();**
  - You can load an HBASE table this way
- **divs = load 'NYSE_dividends' as (name:chararray, ticker:chararray, date, val:float);**
  - Define the data types upfront

PIG - DIAGNOSTIC

explain dataset
Shows Detailed analysis of query

describe dataset
Show Schema

Syntax Highlighting Tools

Eclipse	http://code.google.com/p/pig-eclipse
Emacs	http://github.com/cloudera/piglatin-mode
TextMate	http://www.github.com/kevinweil/pig.tmbundle
Vim	http://www.vim.org/scripts/script.php?script_id=2186

PIG - STORE / DUMP

Store

- Stores the data to a file or other storage
- store processed into 'processed' using PigStorage(',');

Dump

- Prints the value on the screen - print()
- Useful for
  - debugging or
  - using with scripts that redirect output
- Syntax: *dump variable;* (or relation or field name)

PIG - RELATIONAL OPERATIONS - FOREACH

Takes expressions & applies to every record

```
1. divs = load '/users/student/data/NYSE_dividends' as (name, ticker, date, val);
2. avged = foreach divs generate ticker, val
3. dump avged
```

PIG - FOREACH

```
prices = load 'NYSE_daily' as (exchange, symbol, date, open, high, low, close,
                                                    volume, adj_close);
```

Basic Operations such as subtraction/addition

- *gain = foreach prices generate **close - open;***

Using position references

- *gain2 = foreach prices generate **$6 - $3;***

Ranges

- For exchange, symbol, date, open
  - ***beginning = foreach prices generate ..open;***
- For open, high, low, close
  - **middle = foreach prices generate open..close;**
- For produces volume, adj_close
  - ***end = foreach prices generate volume..;***

PIG

Will it have any reducer code generated?
gain = foreach prices generate (close - open);

PIG

Will it have any reducer code generated?
gain = foreach prices generate (close - open);

0

PIG - FOREACH

Extract Data from complex types:

```
bball = load 'baseball' as (name:chararray, team:chararray, traits:bag{}, bat:map[]);
avg = foreach bball generate bat#'batting_average';
```

Tuple Projection

```
A = load 'input' as (t:tuple(x:int, y:int));
B = foreach A generate t.x, t.$1;
```

Bag Projection

```
A = load 'input' as (b:bag{t:(x:int, y:int)});
B = foreach A generate b.x;
```

```
A = load 'input' as (b:bag{t:(x:int, y:int)});
B = foreach A generate b.(x, y);
```

PIG - FOREACH

- divs = load 'NYSE_dividends' as
  - (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
- in_cents = foreach divs generate
  - dividends * 100.0 as dividend, dividends * 100.0;
- describe in_cents;

Built-In Function's List
http://pig.apache.org/docs/r0.13.0/func.html

PIG - FILTER

```
-- filter_matches.pig
divs = load 'NYSE_dividends' as
  (exchange:chararray, symbol:chararray, date:chararray, dividends:float);

Regular Expression
1. startswithcm = filter divs by symbol matches 'CM.*';

Negation
2. notstartswithcm = filter divs by not symbol matches 'CM.*';

Logical Operators
3. result = filter divs by  dividends > 0.10 and dividends < .12;
```

More Operators are are:
http://pig.apache.org/docs/r0.13.0/basic.html

PIG

Will there be any reducer generated?
startswithcm = filter divs by symbol matches 'CM.*';

PIG

Will there be any reducer generated?
startswithcm = filter divs by symbol matches 'CM.*';

No.

PIG - ORDER BY / DISTINCT

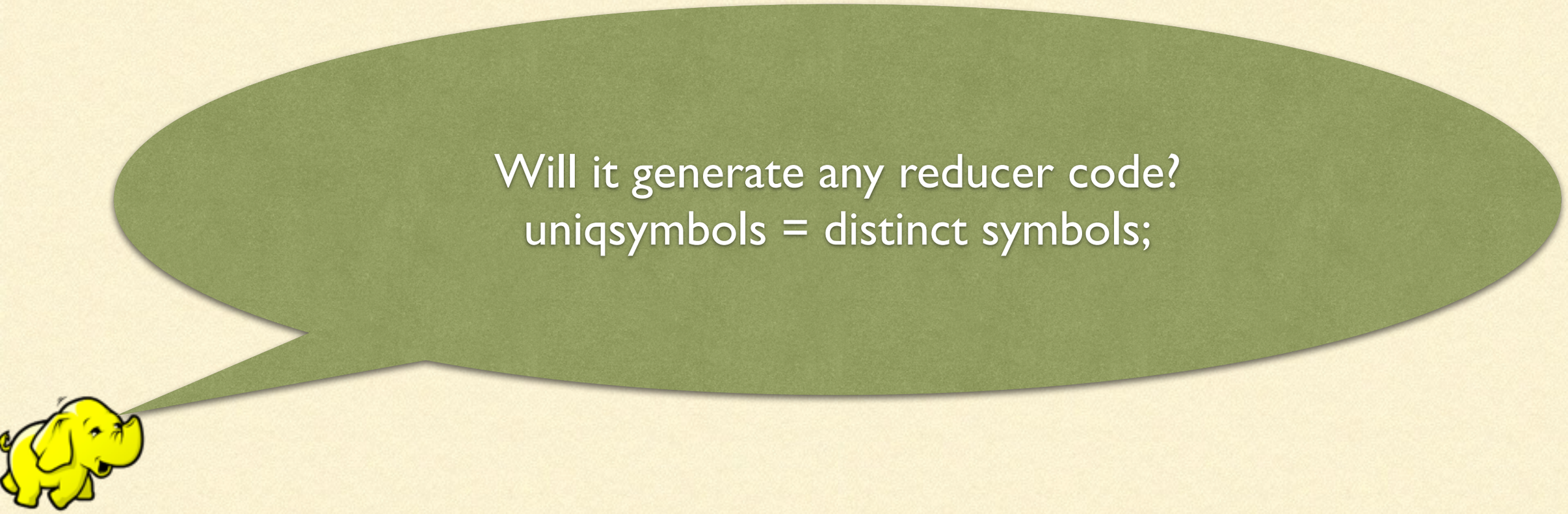
Order BY

```
divs = load 'NYSE_dividends' as
  (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
ordered = order divs by symbol, dividends desc;
```

Distinct

```
divs = load 'NYSE_dividends' as
  (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
symbols = foreach divs generate symbol;
uniqsymbols = distinct symbols;
dump uniqsymbols;
```

PIG


Will it generate any reducer code?
uniqsymbols = distinct symbols;

PIG

Will it generate any reducer code?
uniqsymbols = distinct symbols;

yes

PIG - JOIN

Single Key

```
daily = load 'NYSE_daily' as (exchange, symbol, date, open, high, low, close,volume,
adj_close);
divs  = load 'NYSE_dividends' as (exchange, symbol1, date, dividends);
jnd   = join daily by symbol, divs by symbol1;
```

Composite Key

```
jnd   = join daily by (exchange,symbol), divs by (exchange, symbol);
```

Self Join - For each stock, find dividends that increased between two dates

```
divs1 = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray,
      date:chararray, dividends);
divs2 = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray,
      date:chararray, dividends);
jnd = join divs1 by symbol, divs2 by symbol;
increased = filter jnd by divs1::date < divs2::date and divs1::dividends <
      divs2::dividends;
```

PIG - OUTER JOINS - LEFT

Similar to SQL

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

LO = join *A by id LEFT OUTER*, B by id;
LO = join A by id LEFT, B by id;
dump LO

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

(1,sandeep,1,giri)
(2,ryan,,)

PIG - OUTER JOINS - RIGHT

Similar to SQL

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

```
RO = join A by id RIGHT OUTER, B by id;
or
RO = join A by id RIGHT, B by id;
dump RO;
```

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

(1,sandeep,1,giri)
(,,3,saha)

PIG - OUTER JOINS - FULL

Similar to SQL

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

```
RO = join A by id FULL, B by id;
dump RO;
```

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

```
(1,sandeep,1,giri)
(2,ryan,,)
(,,3,saha)
```

PIG - OUTER JOINS - QUESTION?

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

LO = join A by id, B by id LEFT OUTER;

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

???

PIG - OUTER JOINS - QUESTION?

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

LO = join A by id, B by id LEFT OUTER;

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

ERROR

PIG - OUTER JOINS - QUESTION?

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

```
LO = join A by id OUTER, B by id ;
```

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

PIG - OUTER JOINS - QUESTION?

A

id	*name*
1	sandeep
2	ryan

A = load 'A' as (id, fname);

LO = join A by id OUTER, B by id ;

B

id	*name*
1	giri
3	saha

B = load 'B' as (id, lname);

ERROR

PIG - LIMIT / SAMPLE

LIMIT

```
divs = load 'NYSE_dividends';
first10 = limit divs 10;
dump 10;
```

SAMPLE

```
divs = load 'NYSE_dividends';
some = sample divs 0.1; --10% records
```

PIG

some = sample divs 0.1;
dump some;
dump some;
If you run "dump some" twice, will it give the same output?

PIG

some = sample divs 0.1;
If you run "dump some" twice, will it give the same output?

No. Sampling gives randomised output. It is not like limit.

PIG - PARALLEL

1. Parallelism is the basic promise of pig
2. Basically controls the numbers of reducers
3. Does not impact any mappers

1. daily = load 'NYSE_daily' as (exchange, symbol, date, open, high, low, close, volume, adj_close);
2. bysymbl = group daily by symbol parallel 10;
3. —Will create 10 reducers

1. Needs to be applied per statement
2. Script wide: set default_parallel 10;
3. Without parallel, allocates reducer per 1G of input size

PIG

How many reducers will it produce?
mylist = foreach divs generate symbol, dividends*10

PIG

How many reducers will it produce?
mylist = foreach divs generate symbol, dividends*10

0.

PIG

Does parallel clause have any effect in this?
mylist = foreach divs generate symbol parallel 10;
[Yes/No?]

PIG

Does parallel clause have any effect in this?
mylist = foreach divs generate symbol parallel 10;
[Yes/No?]

N0.

PIG - FLATTEN

```
sandeep  {(c),(c++)}
  arun   {(shell),(c)}
cmaini    {(java),(c++)}
```

```
1. x = load 'myfile' as (name, languages:bag{t:(p)});
2. p1 = foreach x generate name, flatten(languages);
3. dump p1;
```

```
(sandeep,c)
(sandeep,c++)
(arun,shell)
(arun,c)
(cmaini,java)
(cmaini,c++)
```

PIG - CALLING EXTERNAL FUNCTIONS

JAVA Built-IN

```
define hex InvokeForString('java.lang.Integer.toHexString', 'int');
divs  = load 'NYSE_daily' as (exchange, symbol, date, open, high, low,close, volume, adj_close);
inhex = foreach divs generate symbol, hex((int)volume);
```

Your Own Java Function

```
register 'acme.jar';
define convert com.acme.financial.CurrencyConverter('dollar', 'euro');
divs = load 'NYSE_dividends' as (exch:chararray, sym:chararray, date:chararray, vals:float);
backwards = foreach divs generate convert(dividends);
```

Calling a Python function

```
register 'production.py' using jython as bball;
calcprod = foreach nonnull generate name, bball.production(somval);
```

PIG - FOREACH NESTED

blore	sandeep
blore	sandeep
blore	vivek
nyc	john
nyc	john

Find count of distinct names in each city

???

(blore,2)
(nyc,1)

PIG - FOREACH NESTED

```
blore  sandeep
blore  sandeep
blore  vivek
nyc    john
nyc    john
```

Find total distinct names in each city

```
1. people = load 'people' as (city, name);
2. grp = group people by city;
3. counts = foreach grp {
   1. dnames = distinct people.name;
   2. generate group, COUNT(dnames);
4. }
```

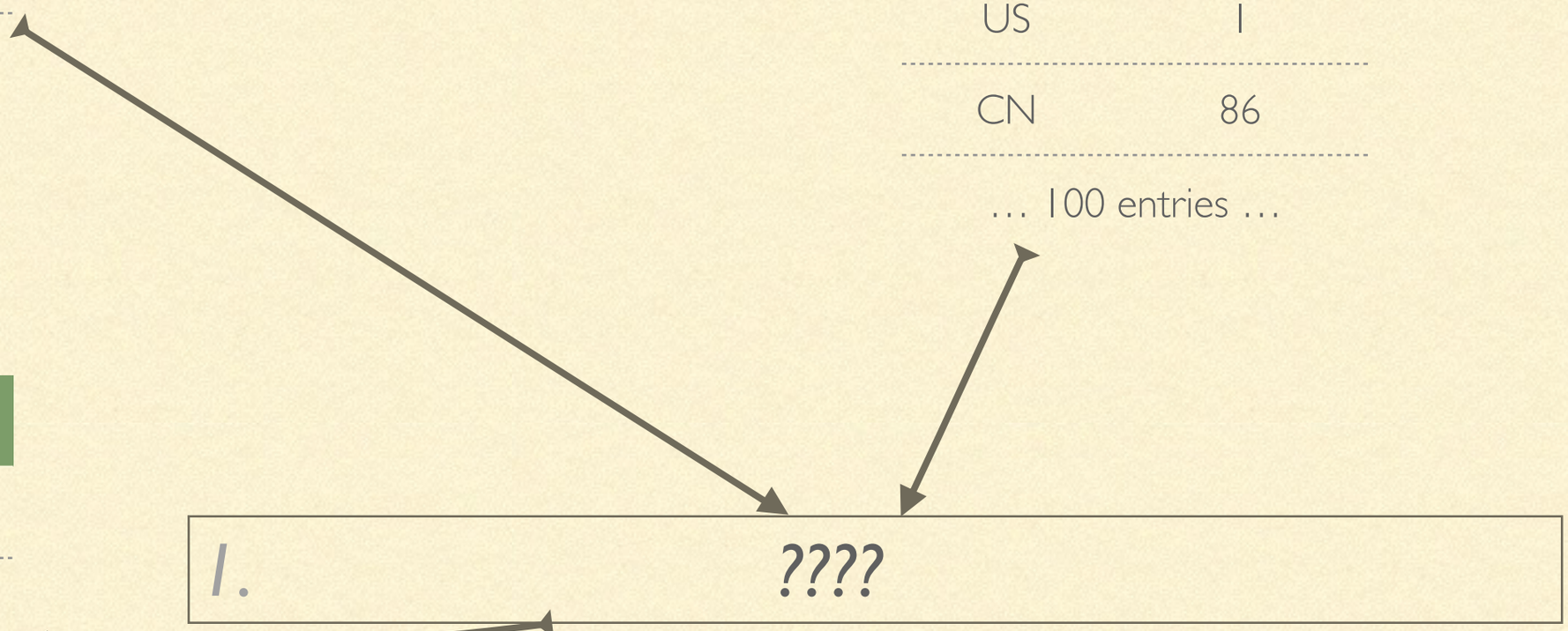
```
(blore,2)
(nyc,1)
```

PIG - FRAGMENTED JOINS

Translate the countries to country code

NAME	COUNTRY
sandeep	IN
kumar	US
gerald	CN
…1 billion entries…	

COUNTY	CODE
IN	91
US	1
CN	86
… 100 entries …	



NAME	CODE
sandeep	91
kumar	1
gerald	86
…1 billion entries…	

PIG - FRAGMENTED JOINS - REPLICATED

Translate the countries to country code

NAME	COUNTRY
sandeep	IN
kumar	US
gerald	CN
…1 billion entries…	

COUNTY	CODE
IN	91
US	1
CN	86
… 100 entries …	

```
1. persons = load 'persons' as (name, country);
2. cc = load 'cc' as (country, code);
3. joined = join persons by country, cc by
   country using 'replicated';
```

NAME	CODE
sandeep	91
kumar	1
gerald	86
…1 billion entries…	

PIG - JOINING SKEWED DATA

Skewed data:

- Too many values for some keys
- Too few keys for some

```
users = load 'users' as (name:chararray, city:chararray);
cinfo = load 'cityinfo' as (city:chararray, population:int);
jnd   = join cinfo by city, users by city using 'skewed';
```

PIG - JOINING SORTED DATA

```
jnd   = join daily by symbol, divs by symbol using 'merge';
```

PIG - COGROUP

Group two tables by a column and then join on the grouped column

owners

person animal

adam cat
adam dog
alex fish
alice cat
steve dog

pets

name animal
nemo fish
fido dog
rex dog
paws cat
wiskers cat

```
COGROUP owners BY animal, pets by animal;
```

```
(cat,{(alice,cat),(adam,cat)},{(wiskers,cat),(paws,cat)})
(dog,{(steve,dog),(adam,dog)},{(rex,dog),(fido,dog)})
(fish,{(alex,fish)},{(nemo,fish)})
```

PIG - UNION

Concatenating two data sets.

Unix Equivalent: cat file1 file2

SQL: union all

```
A = load '/user/me/data/files/input1';
B = load '/user/someoneelse/info/input2';
C = union A, B;
```

TO force schema use

```
A = load '/user/me/data/files/input1' as (x, y);
B = load '/user/someoneelse/info/input2' (x,z);
C = union onschema A, B; -- x,y, z as columns
```

PIG - CROSS

Produce Cross Product

```
tonsodata = cross daily, divs parallel 10;
```

PIG - STREAM - INTEGRATE WITH LEGACY CODE

Say you have a propriety or legacy code that need to applied.

- Invoked once on every map or reduce task
- Not on every record

```
divs = load 'NYSE_dividends' as
    (exchange, symbol, date, dividends);
highdivs = stream divs through `highdiv.pl` as
    (exchange, symbol, date, dividends);
```

The program is shipped to every task node:

```
define hd `highdiv.pl -r xyz` ship('highdiv.pl');
divs = load 'NYSE_dividends' as (exchange, symbol, date, dividends);
highdivs = stream divs through hd as (exchange, symbol, date, dividends);
```

PIG - PARAMETER SUBSTITUTION

Like with any scripting, it is possible to pass an argument
And use it in the script

```
--daily.pig
...
yesterday = filter daily by date == '$DATE';
```

```
pig -p DATE=2009-12-17 daily.pig
```

```
#Param file daily.params
YEAR=2009-
MONTH=12-
DAY=17
DATE=$YEAR$MONTH$DAY
```

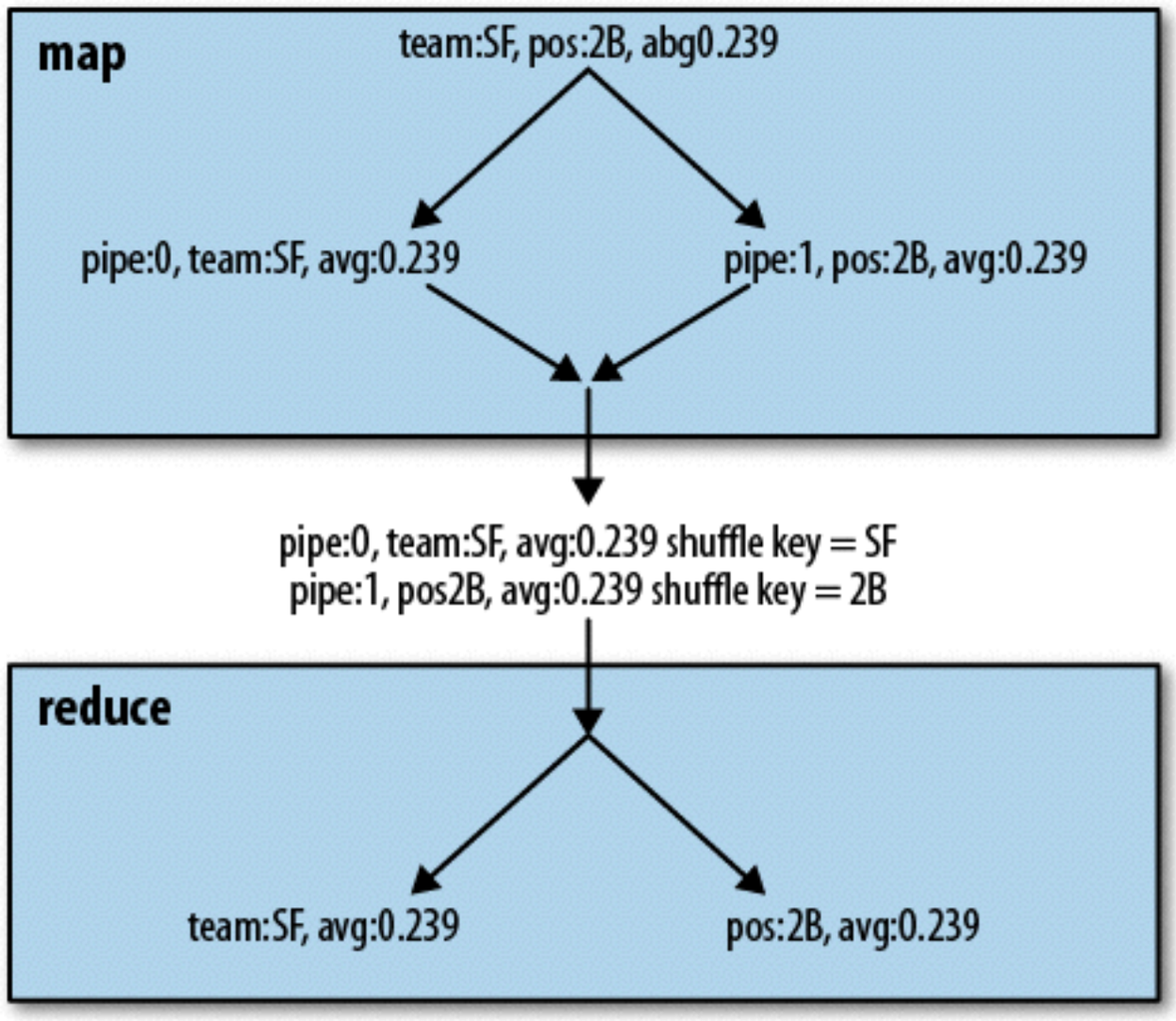
```
pig -param_file daily.params daily.pig
```

PIG - SPLITS

```
wlogs = load 'weblogs' as (pageid, url, timestamp);
split wlogs into
        apr03 if timestamp < '20110404',
        apr02 if timestamp < '20110403' and timestamp > '20110401',
        apr01 if timestamp < '20110402' and timestamp > '20110331';
store apr03 into '20110403';
store apr02 into '20110402';
store apr01 into '20110401';
```

1. A single record can go to multiple legs
2. A record could also be dropped out

PIG - NON LINEAR EXECUTION

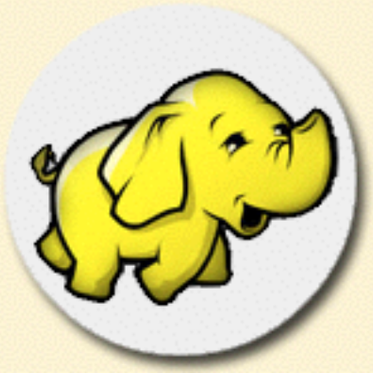


Pig tries to combine multiple groups and filters into one.
And also creates multiple map jobs depending on what we are trying to do.

PIG - REFERENCES

General Docs:
http://pig.apache.org/docs/r0.13.0/

Creating Custom Load Functions:
http://pig.apache.org/docs/r0.13.0/udf.html



Big Data & Hadoop

Thank you.

+91-9538998962

sandeep@knowbigdata.com