Introduction to Pig

Agenda

- What is Pig?
  - Key Features of Pig
- The Anatomy of Pig
- Pig on Hadoop
- Pig Philosophy
- Pig Latin Overview
  - Pig Latin Statements
  - Pig Latin: Identifiers
  - Pig Latin: Comments
- Data Types in Pig
  - Simple Data Types
  - Complex Data Types

Agenda

- Running Pig
- Execution Modes of Pig
- Relational Operators
- Eval Function
- Piggy Bank
- When to use Pig?
- When NOT to use Pig?
- Pig versus Hive

What is Pig?

Apache Pig is a platform for data analysis.

It is an alternative to Map Reduce Programming.

Features of Pig

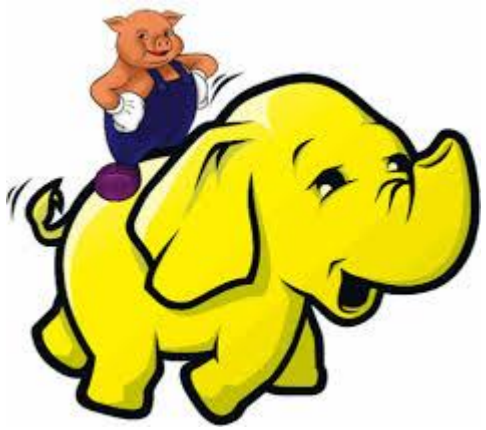
- It provides an **engine** for executing **data flows** (how your data should flow). Pig processes data in parallel on the Hadoop cluster.
- It provides a language called "**Pig Latin**" to express data flows.
- Pig Latin contains operators for many of the traditional data operations such as join, filter, sort, etc.
- It allows users to develop their own functions (User Defined Functions) for reading, processing, and writing data.

The Anatomy of Pig

The main components of Pig are as follows:

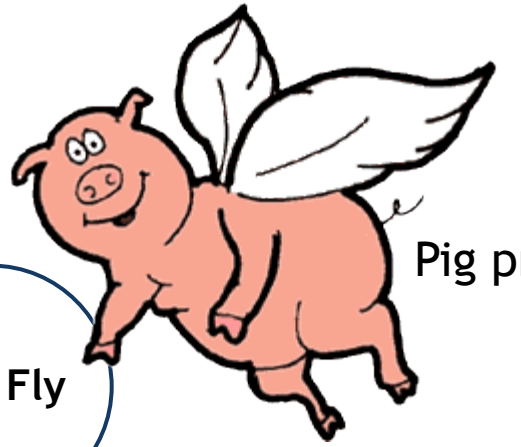
- Data flow language (**Pig Latin**).
  - **Ease of programming.** easy to write, understand, and maintain. (1/20th the lines of code and 1/16th the development time)
  - **Optimization opportunities.** user can focus on semantics rather than efficiency.
  - **Extensibility.** Users can create their own functions to do special-purpose processing.
- Interactive shell where you can type Pig Latin statements (**Grunt**).
- Pig interpreter and execution engine.

Pig on Hadoop



- Pig runs on Hadoop.
- Pig uses both Hadoop Distributed File System and MapReduce Programming.
- By default, Pig reads input files from HDFS. Pig stores the intermediate data (data produced by MapReduce jobs) and the output in HDFS.
- However, Pig can also read input from and place output to other sources.

Pig Philosophy



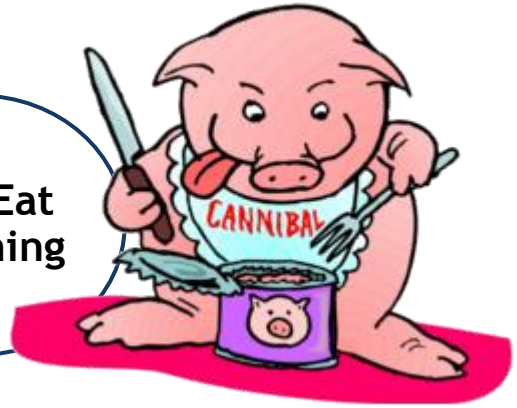
Pig processes data quickly

Pigs Fly

Pigs are Domestic Animals

Pig Philosophy

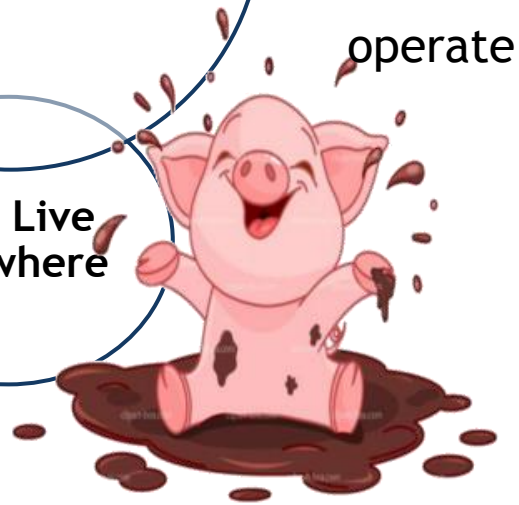
Pigs Eat Anything



easily controlled and modified by its users

operate on relational, nested, or unstructured

Pigs Live Anywhere



Processes files not only from HDFS

Pig Latin Statements

- Pig Latin statements are basic constructs to process data using Pig
- An operator in Pig Latin takes relation as input and yields another relation as output
- Pig Latin statements include schemas and expressions to process data
- Statements end with semicolon

Pig Latin Statements

Pig Latin Statements are generally ordered as follows:

1. **LOAD** statement that reads data from the file system.
2. Series of statements to perform transformations.
3. **DUMP** or **STORE** to display/store result.

```
A = load 'student' (rollno, name, gpa);

A = filter A by gpa > 4.0;

A = foreach A generate UPPER (name);
STORE A INTO ‘myreport’
```

A is a relation

Pig Latin Identifiers

Begins with a letter and followed by letters, numbers and underscores

Valid Identifier	Y	A1	A1_2014	Sample
Invalid Identifier	5	Sales$	Sales%	_Sales

Pig Latin Comments

In Pig Latin two types of comments are supported:

1. Single line comments that begin with “—”.
2. Multiline comments that begin with “/* and end with */”.

Pig Latin Case sensitivity

- Keywords are not case sensitive
- Relations, paths and function names are case sensitive.

Operators in Pig Latin

Arithmetic	Comparison	Null	Boolean
+	= =	IS NULL	AND
-	! =	IS NOT NULL	OR
*	<		NOT
/	>		
%	<=		
	>=		

Data Types in Pig Latin

Simple Data Types

Name	Description
int	Whole numbers
long	Large whole numbers
float	Decimals
double	Very precise decimals
chararray	Text strings
bytearray	Raw bytes
datetime	Datetime
boolean	true or false

Complex Data Types

Name	Description
Tuple	An ordered set of fields. Example: (2,3)
Bag	A collection of tuples. Example: {(2,3),(7,5)}
map	key, value pair (open # Apache)

Running Pig

Pig can run in two ways:

1. Interactive Mode.

2. Batch Mode. (by storing script as .pig)

Execution Modes of Pig

You can execute pig in two modes:

- Local Mode.

  pig –x local filename

- Map Reduce Mode.

  pig filename

hdfs dfs -mkdir /user/cloudera/pigdemo
hdfs dfs -put /home/cloudera/Downloads/students.txt /user/cloudera/pigdemo

Filter

Find the tuples of those student where the GPA is greater than 8.0.

```
A = load '/user/cloudera/pigdemo/student.txt' as (rollno:int, name:chararray, gpa:float);
B = filter A by gpa > 8.0;
DUMP B;
```

FOREACH

Display the name of all students in uppercase.

```
A  =  load '/user/cloudera/pigdemo/student.txt' as (rollno:int, name:chararray, gpa:float);

B = foreach A generate UPPER (name);

DUMP B;
```

Group

Group tuples of students based on their GPA.

```
A  =  load '/user/cloudera/pigdemo/student.txt' as (rollno:int, name:chararray, gpa:float);

B = GROUP A BY gpa;

DUMP B;
```

Distinct

To remove duplicate tuples of students.

```
A  =  load '/user/cloudera/pigdemo/student.txt' as (rollno:int, name:chararray, gpa:float);

B = foreach A generate UPPER (name);

C = DISTINCT B;

DUMP C;
```

Join

To join two relations namely, “student” and “department” based on the values contained in the “rollno” column.

```
A = load '/user/cloudera/pigdemo/students.txt' as (rollno:int, name:chararray, gpa:float);

B = load '/user/cloudera/pigdemo/departments.txt' as (rollno:int, deptno:int,deptname:chararray);

C = JOIN A BY rollno, B BY rollno;

DUMP C;

DUMP B;
```

Split

To partition a relation based on the GPAs acquired by the students.

- GPA = 8.0, place it into relation X.
- GPA is < 8.0, place it into relation Y.

```
A = load '/user/cloudera/pigdemo/students.txt' as (rollno:int, name:chararray, gpa:float);

SPLIT A INTO X IF gpa==8.0, Y IF gpa<8.0;

DUMP X;
```

Avg

To calculate the average marks for each student.

```
A = load '/user/cloudera/pigdemo/students.csv' USING PigStorage (',') as
(studname:chararray,marks:int);

B = GROUP A BY studname;

C = FOREACH B GENERATE A.studname, AVG(A.marks);

DUMP C;
```

Max

To calculate the maximum marks for each student.

```
A = load '/user/cloudera/pigdemo/students.csv' USING PigStorage (',') as
(studname:chararray,marks:int);

B = GROUP A BY studname;

C = FOREACH B GENERATE A.studname, MAX(A.marks);

DUMP C;
```

Map

MAP represents a key/value pair.

To depict the complex data type “map”.

```
John      [city#Bangalore]
Jack      [city#Pune]
James     [city#Chennai]
```

```
A = load '/user/cloudera/pigdemo/studentsmap.txt' Using PigStorage as
(studname:chararray,m:map[chararray]);

 B = foreach A generate m#'city' as CityName:chararray;

DUMP B
```

Word Count using Pig

```
lines = load '/user/cloudera/pigdemo/word_count_text.txt' AS (line:chararray);
words = foreach lines generate flatten(TOKENIZE(line)) as word;
grouped = group words by word;
wordcount = foreach grouped generate COUNT(words), group;
dump wordcount;
store wordcount into '/user/cloudera/pigdemo/pig_wordcount';
```

User defined function

```
package myudfs;

import java.io.IOException;
import org.apache.pig.EvalFunc;
import org.apache.pig.data.Tuple;

public class UPPER extends EvalFunc<String>
{
    public String exec(Tuple input) throws IOException {
        if (input == null || input.size() == 0)
            return null;
        try{
            String str = (String)input.get(0);
           return str.toUpperCase();
        }catch(Exception e){
            throw new IOException("Caught exception processing input row ", e);
        }
    }
}
```

UDF

To use upper function

```
REGISTER /home/cloudera/Desktop/myudfs.jar;

A = LOAD '/user/cloudera/pigdemo/students.txt' as (rollno:int, name:chararray, gpa:float);

B = FOREACH A GENERATE myudfs.UPPER(name);

DUMP B;
```

When to use Pig?

Pig can be used in the following situations:

1. When data loads are time sensitive. - Loading large volumes of data can become a problem as the volume of data increases: the more data there is, the longer it takes to load. Instead of buying faster server, Pig can be used by data professionals to build simple, easily understood scripts to process and analyze massive quantities of data in a massively parallel environment.

2. When processing various data sources. -A job can be written to collect web server logs, use external programs to fetch geo-location data for the users' IP addresses, and join the new set of geo-located web traffic to click maps stored as JSON, web analytic data in CSV format, and spreadsheets from the advertising department to build a rich view of user behavior overlaid with advertising effectiveness.

3. When analytical insights are required through sampling- Pig allows sampling with a random distribution of data. This can reduce the amount of data that needs to be analyzed and still deliver meaningful results.

When NOT to use Pig?

Pig should not be used in the following situations:

1. When data is completely unstructured such as video, text, and audio.

2. When there is a time constraint because Pig is slower than MapReduce jobs. (eg: cross join)

Are you better than the Pig optimizer than figuring out how to string multiple jobs together (and other things)?

Pig Vs. Hive

Features	Pig	Hive
Used By	Programmers and Researchers	Analyst
Used For	Programming	Reporting
Language	Procedural data flow language	SQL Like
Suitable For	Semi - Structured	Structured
Schema / Types	Explicit	Implicit
UDF Support	YES	YES
Join / Order / Sort	YES	YES
DFS Direct Access	YES (Explicit)	YES (Implicit)
Web Interface	No	Yes (HWI)
Partitions	No	Yes
Shell	YES	YES

Answer a few quick questions …

Fill in the blanks

1. Pig is a _____ language.
2. In Pig, _____ is used to specify data flow.
3. Pig provides an _____ to execute data flow.
4. _____, _____ are execution modes of Pig.
5. Pig is used in _____ process.

References …

Further Readings

- http://pig.apache.org/docs/r0.12.0/index.html
- http://www.edureka.co/blog/introduction-to-pig/

Thank you