Introduction to Pig

What is Pig?

- Apache pig is a platform for data analysis.
- It is an alternative to MapReduce programming
- It was developed as a research project at Yahoo.

Why Do We Need Apache Pig?

- Programmers who are not so good at Java
- Apache Pig is a boon for all such programmers.
- Using Pig Latin, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses multi-query approach, thereby reducing the length of codes.
- Pig Latin is SQL-like language and it is easy to learn Apache Pig when you are familiar with SQL.

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

- Data flow language(Pig Latin)
- Interactive shell where you can type Pig Latin statements(grunt)
- Pig interpreter and execution engine.

Pig Latin

Pig Latin Script

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

A=filter A by gpa >4;

A=foreach A generate UPPER (name);

STORE A INTO 'myreport';
```

Pig Interpreter/Execution Engine

- Processes and parses pig Latin
- Checks data types
- Performs optimization
- Creates MapReduce job
- Submits jobs to Hadoop
- Monitor progress

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 On Hadoop

- Pig Supports the following:

- HDFS commands
- UNIX shell commands
- Relational operators
- Positional parameters
- Common mathematical functions
- Custom functions
- Complex data structures

Pig Philosophy

- **Pigs Eat Anything: Pig** can process different kinds of data such as structured and unstructured data.
- **Pigs Live Anywhere**: Pig not only processes files in HDFS, it also process other sources such as files in the local file system.
- **Pigs are Domestic Animals**: Pig allows you to develop user define functions and the same can be included in the script for complex operations
- **Pigs fly**: Pig process data quickly.

Apache Pig Vs MapReduce

Apache Pig	MapReduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid.
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

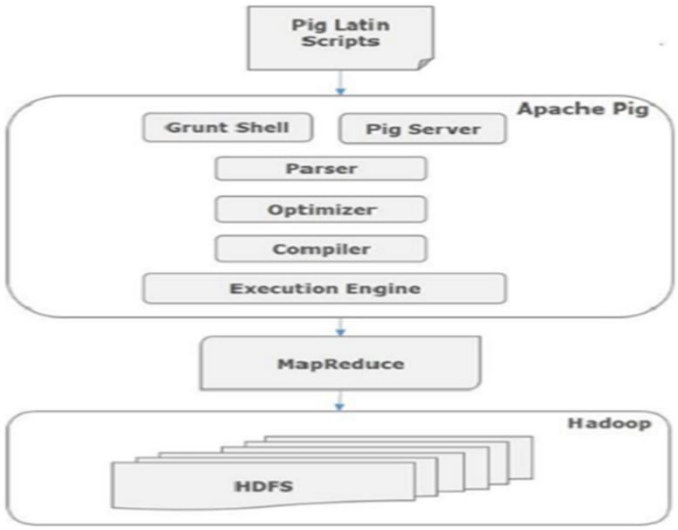
Apache Pig Vs SQL

Pig	SQL
Pig Latin is a procedural language.	SQL is a declarative language.
In Apache Pig, schema is optional. We can store data without designing a schema (values are stored as $01, $02 etc.)	Schema is mandatory in SQL.
The data model in Apache Pig is nested relational.	The data model used in SQL is flat relational.
Apache Pig provides limited opportunity for Query optimization.	There is more opportunity for query optimization in SQL.

Apache Pig vs Hive

Pig	Hive
Apache Pig uses a language called Pig Latin. It was originally created at Yahoo.	Hive uses a language called HiveQL. It was originally created at Facebook.
Pig Latin is a data flow language.	HiveQL is a query processing language
Pig Latin is a procedural language and it fits in pipeline paradigm.	HiveQL is a declarative language.
Apache Pig can handle structured, unstructured, and semi-structured data.	Hive is mostly for structured data.

Pig Architecture

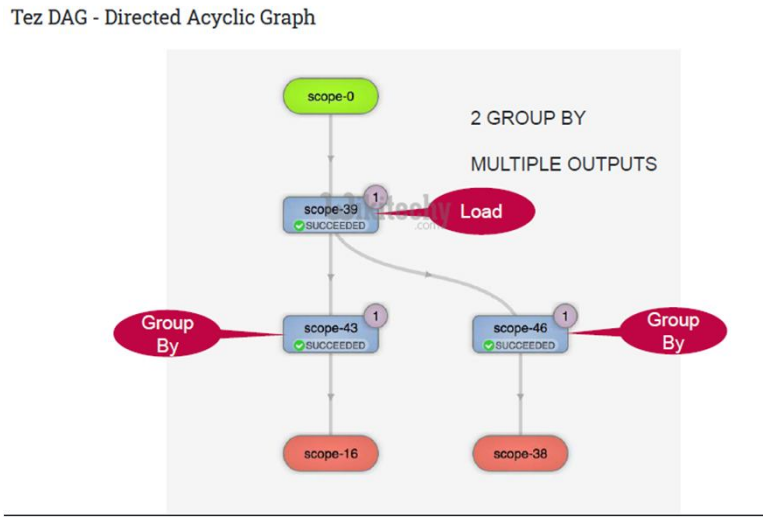


Pig architecture

- **Parser**
  - Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks.
  - The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.
  - In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.
- **Optimizer**
  - The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.
- **Compiler**

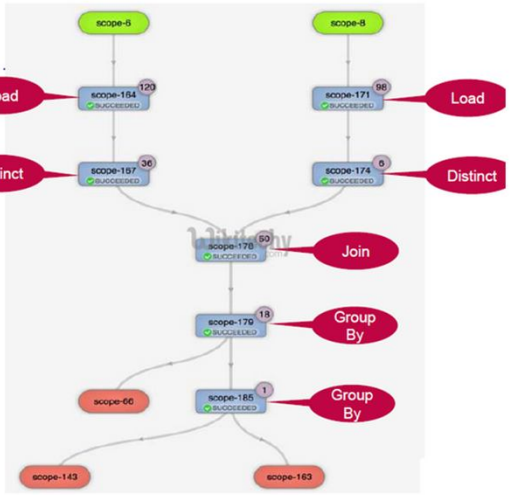
  The compiler compiles the optimized logical plan into a series of MapReduce jobs.
- **Execution engine**

  Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

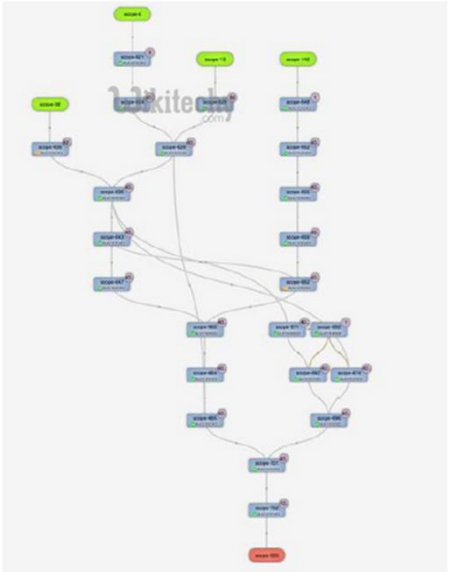


- 2 DISTINCT + JOIN + 2 GROUP BY

- Multiple inputs
- Multiple HDFS outputs



High Depth DAG - Directed Acyclic Graph



Pig Latin statements

- Pig Latin statements are basic constructs to process data using Pig
- Pig Latin statements is an operator
- An operator in Pig Latin takes a relation as input and yields another relation as output.
- Pig Latin statements include schemas and expressions to process data
- Pig Latin statements should end with a semi colon.

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 not a variable

Pig Latin: Keywords

- Keyword are reserved.
- It cannot be used to name anything

Pig Latin : Identifiers

- Identifiers are names assigned to fields or other data structures.
- It should begin with a letter and should be followed only 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:

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

Pig Latin: Case sensitive

- Keywords are not case sensitive such as LOAD, STORE,GROUP FOREACH etc.
- Relations and paths are case sensitive
- Function names are case sensitive such as PigStorage, COUNT

Operators in Pig Latin

Operator	Description
+	**Addition** – Adds values on either side of the operator
–	**Subtraction** – Subtracts right hand operand from left hand operand
*	**Multiplication** – Multiplies values on either side of the operator
/	**Division** – Divides left hand operand by right hand operand
%	**Modulus** – Divides left hand operand by right hand operand and returns remainder

Data Types in Pig Latin

Simple data types

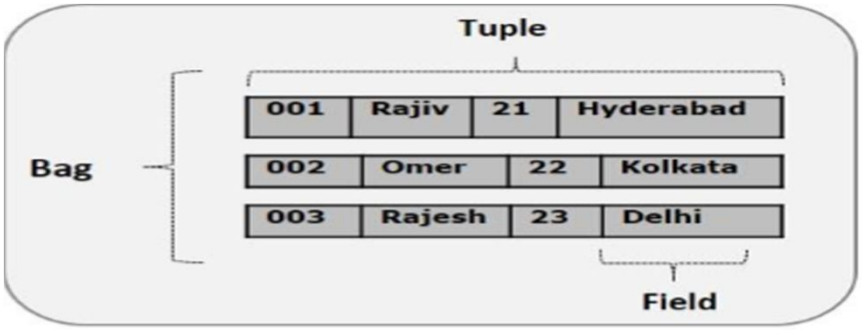
Types	Description	Example
int	Signed 32 bit Integer	10
long	Signed 64 bit Integer	10L, 10l
float	32 bit floating point	10.5F, 10.5f
double	64 bit floating point	10.5
chararray	String in UTF-8	Hello World
bytearray	Binary array	-
boolean	Boolean	true/false (case insensitive)
datetime	Date and time in ISO 8601 format – YYYY-MM-DDThh:mm:ss.sTZD	1997-07-16T19:20:30.45+01:00

Pig Latin:Data types

- Complex Data Types

Data Type	Description	Example
tuple	An ordered set of fields.	(1,2,6)
bag	A collection of tuples.	{(1,2,6), (3,4,5)}
map	A set of key value pairs.	[name#Serkan,country#US]

Complex data type



Complex Data Type:Tuple

Tuple
A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.

Complex Data type:BAG

- A bag is an unordered set of tuples.
- In other words, a collection of tuples (non-unique) is known as a bag.
- Each tuple can have any number of fields (flexible schema).
- A bag is represented by '{}'. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.
- **Example** – {(Raja, 30), (Mohammad, 45)}

Complex Data type: **MAP and Relation**

Map
A map (or data map) is a set of key-value pairs.

The **key** needs to be of type chararray and should be unique.

The **value** might be of any type.

Relation
A relation is a bag of tuples. The relations in Pig Latin are unordered

Running Pig

Pig can run in two ways:

Interactive Mode: By invoking **GRUNT** shell

Batch Mode: Write Pig Latin in File save it with .pig extension

Execution Modes of Pig

You can execute pig in two modes:

Local Mode.

Keep your files in local file system.

pig –x Local file name

Map reduce

Need access Hadoop cluster to read/write file.

Default mode of pig

Pig File name

Relational Operators

Relational operators

Operator	Description
	Loading and Storing
LOAD	To Load the data from the file system (local/HDFS) into a relation.
STORE	To save a relation to the file system (local/HDFS).
	Filtering
FILTER	To remove unwanted rows from a relation.
DISTINCT	To remove duplicate rows from a relation.
FOREACH, GENERATE	To generate data transformations based on columns of data.
STREAM	To transform a relation using an external program.
	Grouping and Joining
JOIN	To join two or more relations.
COGROUP	To group the data in two or more relations.
GROUP	To group the data in a single relation.
CROSS	To create the cross product of two or more relations.

SORTING

	Sorting
ORDER	To arrange a relation in a sorted order based on one or more fields (ascending or descending).
LIMIT	To get a limited number of tuples from a relation.
	Combining and Splitting
UNION	To combine two or more relations into a single relation.
SPLIT	To split a single relation into two or more relations.
	Diagnostic Operators
DUMP	To print the contents of a relation on the console.
DESCRIBE	To describe the schema of a relation.
EXPLAIN	To view the logical, physical, or MapReduce execution plans to compute a relation.
ILLUSTRATE	To view the step-by-step execution of a series of statements.

STORAGE FUNCTION

- The **PigStorage()** function loads and stores data as structured text files.
- It takes a delimiter using which each entity of a tuple is separated as a parameter. By default, it takes '**\t'** as a parameter.
- Pig Storage is a built-in function of Pig, and one of the most common functions used to load and store data in pig scripts. Pig Storage can be used to parse text data with an arbitrary delimiter, or to output data in an delimited format.

Example

```
grunt> A = load '/home/hduser/Desktop/student.csv' USING
PigStorage(',') as (rno:int,name:chararray,gpa:float);

grunt> B = Group A by name;

grunt> c = FOREACH B Generate A.name,AVG(A.gpa);

grunt> DUMP c;
```

Filter

Find the tuples of those student where the GPA is greater than 4.0.

```
A = load '/pigdemo/student.tsv' as (rollno:int,
name:chararray, gpa:float);
B = filter A by gpa > 4.0; DUMP B;
```

FOREACH

Display the name of all students in uppercase.

```
A = load '/pigdemo/student.tsv' 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 '/pigdemo/student.tsv' as (rollno:int, name:chararray, gpa:float);
B = GROUP A BY gpa; DUMP B;
```

COGROUP

```
grunt> cogroup_data = COGROUP student_details by age, employee_details by age;
```

Distinct

To remove duplicate tuples of students.

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

B = DISTINCT A;

DUMP B;
```

Limit

Limit the number of output tuples

Display the first three tuples from "student" relation

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

B = LIMIT A 3;

DUMP B;
```

ORDER BY

Used to sort relation

Display the names of the students in Ascending order.

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

B = ORDER A BY name;

DUMP B;
```

Join

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

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

B = load '/pigdemo/department.tsv' as (rollno:int,

deptno:int,deptname:chararray);  C = JOIN A BY rollno, B

BY rollno;

DUMP C;  DUMP B;
```

UNION

It is used to merge the contents of two relations

To merge two relations "student" and "department"

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

B = load '/pigdemo/department.tsv' as (rollno:int,

deptno:int,deptname:chararray);  C = UNION A,B;

STORE C INTO '/pigdemos/uniondemo';

DUMP C;
```

Split

To partition a relation based on the GPAs acquired by the students.
- GPA = 4.0, place it into relation X.
- GPA is < 4.0, place it into relation Y.

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

SPLIT A INTO X IF gpa==4.0, Y IF gpa<=4.0;

DUMP X;
```

SAMPLE

Used to select random sample of data based on the specified sample size

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

B = SAMPLE A 0.01;

DUMP B;
```

Eval Function

Avg

To calculate the average marks for each student.

```
A = load '/pigdemo/student.csv' USING PigStorage (‘,’) as
(studname:chararray,marks:int);

B = GROUP A BY studname;

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

DUMP C;
```

Pig Storage is a built-in function of Pig, and one of the most common functions used to load and store data in pig scripts. Pig Storage can be used to parse text data with an arbitrary delimiter, or to output data in an delimited format.

Max

To calculate the maximum marks for each student.

```
A = load '/pigdemo/student.csv' USING PigStorage (‘,’) as

(studname:chararray, marks:int);  B = GROUP A BY studname;

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

DUMP C;
```

COUNT

used to count the number of elements in a bag

```
A = load '/pigdemo/student.csv' USING PigStorage (‘,’) as (studname:chararray,marks:int);

B = GROUP A BY studname;

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

DUMP C;
```

COMPLEX DATA TYPES

TUPLE

Tuple is an ordered collections of fields

INPUT:

(x,12) (y,13)
(z,7) (f,5)
(s,8) (sc,12)

```
A=load '/root/pigdemos/studentdata.tsv' AS

(t1:tuple(t1a:chararray,t1b:int),t2:tuple(t2a:chararray,t2b:int));

B= FOREACH A GENERATE t1.t1a,t1.t1b,t2.$0,t2.$1;

DUMP B;
```

Output:

(x,12,y,13)
(z,7,f,5)
(s,8,sc,12)

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 '/root/pigdemos/studentcity.tsv' Using PigStorage as
(studname:chararray,m:map[chararray]);

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

DUMP B
```