Apache - PIG

Ques: What is Apache Pig?

It is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger set of data representing them as data flows. Pig is generally used with Hadoop, we can perform all the data manipulation operations in Hadoop using Apache Pig

To write data analysis programs, Pig provides a high level language known as Pig Latin. This language provides various operators using which programmers can develop their own functions for reading, writing and processing data.

To analyze data using Apache Pig, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into Map Reduce jobs.

Ques: Why do we need Apache Pig?

Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing MapReduce task. Apache Pig is a boon for all such programmers.

- Using Pig Latin, programmers can perform MapReduce task easily without having to type complex codes in Java.
- ★ Apache Pig uses multi-query approach, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code LoC in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- ★ Pig Latin is SQL-like language and it is easy to learn Apache Pig when you are familiar with SQL.
- ★ Apache Pig provides many built-in operators to support data operations like joins, filters, ordering etc. In addition, it also provides nested data types like tuples, bags and maps that are missing from Map Reduce.

: Features of Pig.

Apache Pig comes with the following features -

- Rich set of operators - It provides many operators to perform operations like join, sort, filter etc.
- Ease of programming - Pig Latin is similar to SQL and its is easy to write a Pig script if you are good at SQL
- Optimization opportunities - The task in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
- Extensibility - Using the existing operators, users can develop their own functions to read, process and write data.
- UDF's - Pig provides the facility to create User-defined Functions in other programming languages such as Java and invoke or embed them in Pig scripts.
- Handles all kinds of data - Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Ques. Apache Pig Vs. MapReduce.

Apache Pig	Map Reduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level languages.	It is low level and rigid.
~~Progr~~ Performing a join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce

ache Pig uses multi-uery approach, thereby educing the length of the ode 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.

Ques: Apache Pig Vs. Hive

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

Ques: Application of Apache Pig

Apache Pig is generally used by data scientists for performing tasks involving ad-hoc processing and quick prototyping. Apache Pig is used.

★ To process huge data sources such as web logs.

To perform data processing for search platforms.

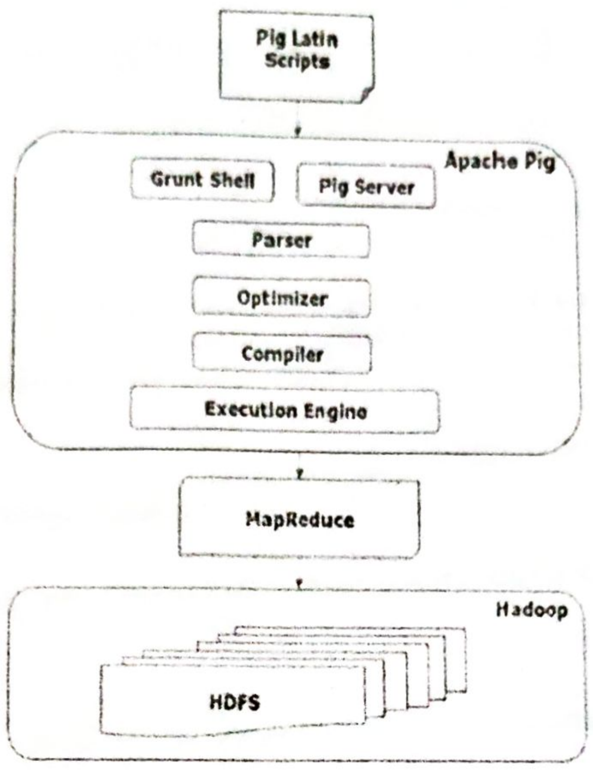
* To process time sensitive data loads.

Apache Pig - Architecture

The language used to analyze data in Hadoop using Pig is known as **Pig Latin**. It is a high level data processing language which provides a rich set of data types and operators to perform various operations on the data.

To perform a particular task Programmers using Pig, programmers need to write a Pig script using the Pig Latin language, and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded). After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output.

Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy. The architecture of Apache Pig is shown below.



Apache Pig Components

As shown in the figure, there are various components in the Apache Pig framework. Let us take a look at the major components.

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

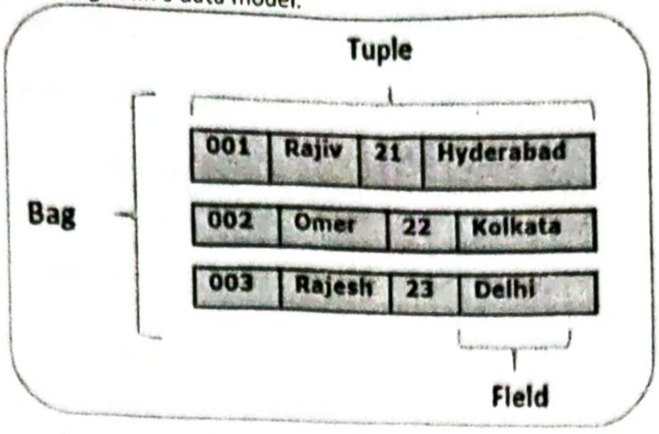
The compiler compiles the optimized logical plan into a series of MapReduce jobs.

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.

g Latin Data Model

he data model of Pig Latin is fully nested and it allows complex non-atomic datatypes such as **map** and **tuple**. Given below is the diagrammatical representation of Pig Latin's data model.



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)}

A bag can be a field in a relation; in that context, it is known as **inner bag**.

Example – {Raja, 30, **{9848022338, raja@gmail.com,}**}

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. It is represented by '[]'

Example – [name#Raja, age#30]

Relation

A relation is a bag of tuples. The relations in Pig Latin are unordered (there is no guarantee that tuples are processed in any particular order).

Apache Pig - Grunt Shell

After invoking the Grunt shell, you can run your Pig scripts in the shell. In addition to that, there are certain useful shell and utility commands provided by the Grunt shell. This chapter explains the shell and utility commands provided by the Grunt shell.

Note – In some portions of this chapter, the commands like **Load** and **Store**are used. Refer the respective chapters to get in-detail information on them.

Shell Commands

The Grunt shell of Apache Pig is mainly used to write Pig Latin scripts. Prior to that, we can invoke any shell commands using **sh** and **fs**.

sh Command

Using **sh** command, we can invoke any shell commands from the Grunt shell. Using **sh** command from the Grunt shell, we cannot execute the commands that are a part of the shell environment (**ex** – cd).

Syntax

Given below is the syntax of **sh** command.

```
grunt> sh shell command parameter
```

fs Command

Using the **fs** command, we can invoke any FsShell commands from the Grunt shell.

Syntax

Given below is the syntax of **fs** command.

```
grunt> sh File System command parameters
```

tility Commands

The Grunt shell provides a set of utility commands. These include utility commands such as **clear, help, history, quit,** and **set;** and commands such as **exec, kill,** and **run** to control Pig from the Grunt shell. Given below is the description of the utility commands provided by the Grunt shell.

clear Command

The **clear** command is used to clear the screen of the Grunt shell.

Syntax

You can clear the screen of the grunt shell using the **clear** command as shown below.

```
grunt> clear
```

history Command

This command displays a list of statements executed / used so far since the Grunt sell is invoked.

set Command

The **set** command is used to show/assign values to keys used in Pig.

quit Command

You can quit from the Grunt shell using this command.

Usage

Quit from the Grunt shell as shown below.

```
grunt> quit
```

exec Command

Using the **exec** command, we can execute Pig scripts from the Grunt shell.

Syntax

Given below is the syntax of the utility command **exec**.

```
grunt> exec [-param param_name = param_value] [-param_file file_name] [script]
```

kill Command

You can kill a job from the Grunt shell using this command.

Syntax

Given below is the syntax of the **kill** command.

```
grunt> kill JobId
```

run Command

You can run a Pig script from the Grunt shell using the **run** command

Syntax

Given below is the syntax of the **run** command.

```
grunt> run [-param param_name = param_value] [-param_file file_name] script
```

it is [illegible] as a [illegible]

PIG LATIN

Pig Latin - Data Model

The data model of Pig is fully nested. A Relation is the outermost structure of the Pig Latin data model. And it is a bag where –

- A bag is a collection of tuples.
- A tuple is an ordered set of fields.
- A field is a piece of data.

Pig Latin - Statements.

While processing data using Pig Latin, statements are the basic constructs.

These statements work with relations. They include expressions and schemas.

* Every statement ends with a semicolon ;.
* We will perform various operations using operators provided by Pig Latin, through statements.
* Except LOAD and STORE, while performing all other operations, Pig Latin statements take a relation as input and produce another relation as output.
* As soon as you enter a Load statement in the Grunt shell, its semantic checking will be carried out. To see the contents of the schema you need to use the Dump operator. Only after performing the dump operation, the Map Reduce job for loading the data into file system will be carried out.

Pig Latin – Data types

Given below table describes the Pig Latin data types.

S.N.	Data Type	Description & Example
1	int	Represents a signed 32-bit integer. **Example** : 8
2	long	Represents a signed 64-bit integer. **Example** : 5L
3	float	Represents a signed 32-bit floating point. **Example** : 5.5F
4	double	Represents a 64-bit floating point. **Example** : 10.5
5	chararray	Represents a character array (string) in Unicode UTF-8 format. **Example** : 'tutorials point'
6	Bytearray	Represents a Byte array (blob).
7	Boolean	Represents a Boolean value. **Example** : true/ false.
8	Datetime	Represents a date-time. **Example** : 1970-01-01T00:00:00.000+00:00
9	Biginteger	Represents a Java BigInteger. **Example** : 60708090709
10	Bigdecimal	Represents a Java BigDecimal **Example** : 185.98376256272893883
Complex Types		
11	Tuple	A tuple is an ordered set of fields. **Example** : (raja, 30)
12	Bag	A bag is a collection of tuples. **Example** : {(raju,30),(Mohhammad,45)}
13	Map	A Map is a set of key-value pairs. **Example** : ['name'#'Raju', 'age'#30]

Null Values

Values for all the above data types can be NULL. Apache Pig treats null values in a similar way as SQL does. A null can be an unknown value or a non-existent value. It is used as a placeholder for optional values. These nulls can occur naturally or can be the result of an operation.

Pig Latin – Arithmetic Operators

The following table describes the arithmetic operators of Pig Latin. Suppose a = 10 and b = 20.

Operator	Description	Example
+	Addition – Adds values on either side of the operator	a + b will give 30
-	Subtraction – Subtracts right hand operand from left hand operand	a – b will give –10
*	Multiplication – Multiplies values on either side of the operator	a * b will give 200
/	Division – Divides left hand operand by right hand operand	b / a will give 2
%	Modulus – Divides left hand operand by right hand operand and returns remainder	b % a will give 0
? :	Bincond – Evaluates the Boolean operators. It has three operands as shown below. variable x = (expression) ? value1 if true : value2 if false.	b = (a == 1)? 20: 30; if a = 1 the value of b is 20. if a!=1 the value of b is 30.
CASE WHEN THEN ELSE END	Case – The case operator is equivalent to nested bincond operator.	CASE f2 % 2 WHEN 0 THEN 'even' WHEN 1 THEN 'odd' END

Pig Latin – Comparison Operators

The following table describes the comparison operators of Pig Latin.

Operator	Description	Example
==	Equal – Checks if the values of two operands are equal or not; if yes, then the condition becomes true.	(a = b) is not true
!=	Not Equal – Checks if the values of two operands are equal or not. If the values are not equal, then condition becomes true.	(a != b) is true.
>	Greater than – Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.	(a > b) is not true.
<	Less than – Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.	(a < b) is true.
>=	Greater than or equal to – Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.	(a >= b) is not true.

<=	Less than or equal to – Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.	(a <= b) is true.
matches	Pattern matching – Checks whether the string in the left-hand side matches with the constant in the right-hand side.	f1 matches '.*tutorial.*'

Pig Latin – Type Construction Operators

The following table describes the Type construction operators of Pig Latin.

Operator	Description	Example
()	Tuple constructor operator – This operator is used to construct a tuple.	(Raju, 30)
{}	Bag constructor operator – This operator is used to construct a bag.	{(Raju, 30), (Mohammad, 45)}
[]	Map constructor operator – This operator is used to construct a tuple.	[name#Raja, age#30]