Unit V : Frameworks

Applications on Big Data Using Pig and Hive – Data processing operators in Pig – Hive services – HiveQL – Querying Data in Hive - fundamentals of HBase and ZooKeeper - IBM InfoSphere BigInsights and Streams. Visualizations - Visual data analysis techniques, interaction techniques; Systems and applications

Apache Pig

- It is a tool/platform which is used to analyze larger sets of data representing them as data flows.
- Hadoop Pig is nothing but an abstraction over MapReduce.
- Pig is made up of two pieces:
  • The language used to express data flows, called Pig Latin.
  • **The execution environment to run Pig Latin programs. There are currently two environments: local execution in a single JVM and distributed execution on a Hadoop cluster.**
- Pig is a scripting language for exploring large datasets.

Features of Pig

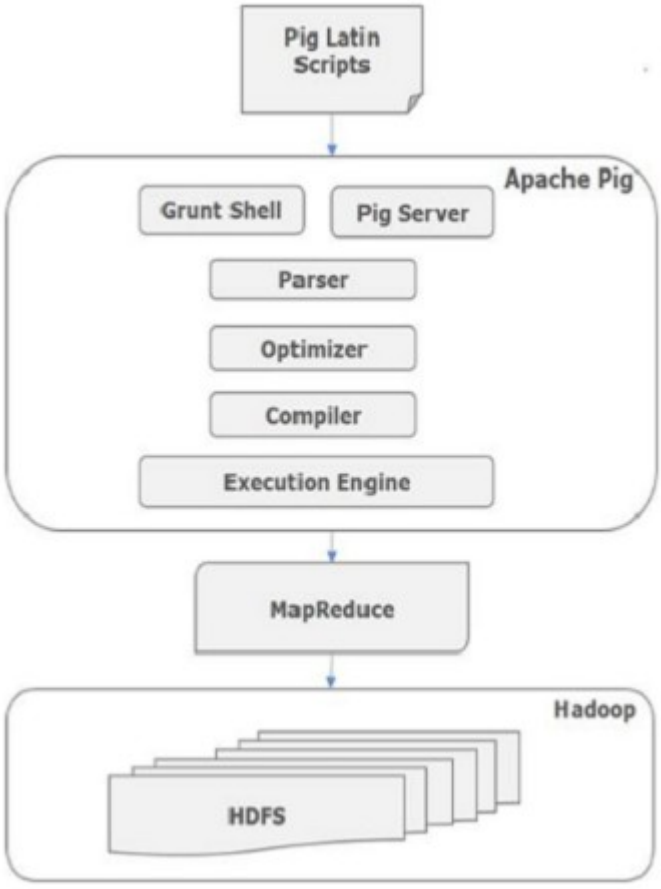
- Rich set of operators
  - It provides many operators to perform operations like join, sort, filer, etc.
- Ease of programming
  - Ability to process terabytes of data simply by issuing a half-dozen lines of Pig Latin from the console.

- Optimization opportunites
  - It allows users to focus on semantics rather than efficiency, to optimize their execution automatically.
- Extensibility
  - In order to do special-purpose processing, users can create their own functions.
  - These functions are converted internally into Map and reduce tasks.
- User defined Functions
  - 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.

Why Pig?

- 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. 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 MapReduce.

Architecture of Hadoop Pig



The major components in Pig framework are:

- **Parser**
  - Parser checks the syntax of the script and other miscellaneous checks.
  - Parser's output will be a DAG (directed acyclic graph). That represents the Pig Latin statements as well as logical operators.
  - The logical operators of the script are represented as the nodes and the data flows are represented as edges, in the DAG (the logical plan).

- **Optimizer**
  - DAG is passed to the logical optimizer.
  - Optimizer carries out the logical optimizations, projection and push down.
- **Compiler**
  - The compiler compiles the optimized logical plan into a series of MapReduce jobs.

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

Pig Vs MapReduce

Pig	MapReduce
Data flow language	Data processing paradigm
High-level language	Low-level
Programmer's SQL knowledge is enough	Programmer needs exposure to Java
Length of code is less	20 times more
Compilation is not required	Compilation process is long

Pig Vs SQL

Pig	SQL
Procedural Language	Declarative Language
Schema is optional	Schema declaration is essential
Nested Relational model	Flat relational model

Apache Pig Vs Hive

Pig	Hive
Language is PigLatin. Created at Yahoo.	Language is HiveQL.Created at Facebook
Data flow language	Query processing language
Procedural language	Decalarative language
Handle structured, unstructured and semi-strucutred data.	Most for strucutred data.

Applications of Pig

- For performing tasks involving ad-hoc processing.
- To process huge data sources like weblogs.
- To perform data processing for search platforms.
- To process time sensitive data loads.

Pig Execution

- Prerequisites
  - Hadoop and Java JDK installed.
- Download Pig

- Download a stable release from *http://hadoop.apache.org/pig/releases.html*
- Unpack the tarball in a suitable place on your workstation:

```
% tar xzf pig-x.y.z.tar.gz
```

- It's convenient to add Pig's binary directory to your command-line path.

```
% export PIG_INSTALL=/home/tom/pig-x.y.z
% export PATH=$PATH:$PIG_INSTALL/bin
```

Execution Types

- **Local Mode**
  - All the files are installed and run from your local host and local file system.
  - There is no need of Hadoop or HDFS.
  - This mode is generally used for testing purpose.

- **Hadoop Mode**
  - Pig translates queries into MapReduce jobs and runs them on a Hadoop cluster.
  - The cluster may be a pseudo- or fully distributed cluster.
  - This mode is mainly used to run Pig on large datasets.

Apache Pig Execution Mechanisms

Apache Pig scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode.

- **Interactive Mode** (Grunt shell) − You can run Apache Pig in interactive mode using the Grunt shell. In this shell, you can enter the Pig Latin statements and get the output (using Dump operator).
- **Batch Mode** (Script) − You can run Apache Pig in Batch mode by writing the Pig Latin script in a single file with **.pig** extension.
- **Embedded Mode** (UDF) − Apache Pig provides the provision of defining our own functions (**U**ser **D**efined **F**unctions) in programming languages such as Java, and using them in our script.

Grunt

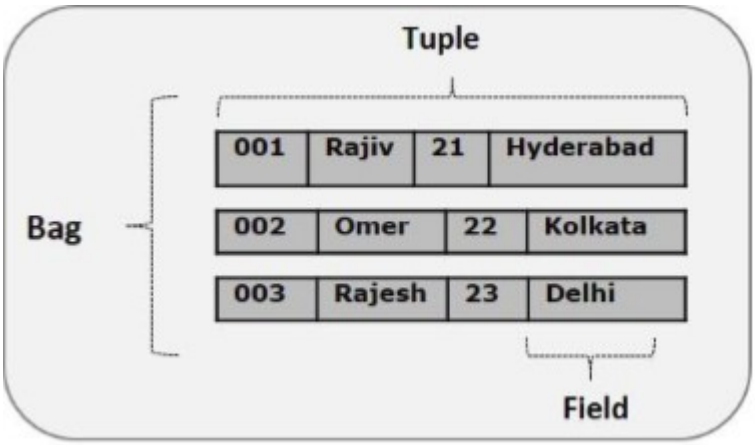
- Grunt is an interactive shell for running Pig commands.
- Grunt is started when no file is specified for Pig to run.
- Grunt offers many shell and utility commands.
- The command sh is used to invoke any shell from Grunt shell.
- Any fs Shell commands from the Grunt shell by using the fs command.

Pig Latin

- Pig Latin is the language used to analyze data in Hadoop.
- The data model of Pig is fully nested.
- A **Relation** is the outermost structure of the Pig Latin data model.

- A relation is a **bag,** where,
  - A bag is a collection of tuples. **Example** − {(Raja, 30), (Mohammad, 45)}
  - A tuple is an ordered set of fields. **Example** − (Raja, 30)
  - A field is a piece of data.Map. **Example** − ‘raja’ or ‘30’
  - 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]



Statements in Pig Latin

- Statements work with relations.
- A statement includes expressions and schemas.
- Every statement ends with a semicolon (;).
- Pig Latin statements take a relation as input and produce another relation as output.
- Its semantic checking will be carried out, once we enter a Load statement in the Grunt shell.

Given below is a Pig Latin statement, which loads data to Apache Pig.

```
grunt>Student_data==LOAD'student_data.txt' USING PigStorage[1](',')as
   ( id:int, firstname:chararray, lastname:chararray, phone:chararray,
city:chararray );
```

Pig Latin – Data types

1 **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.

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

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]

Pig Latin Operators

Arithmetic Operators	
+	Addition
-	Subtraction
*****	Multiplication
/	Division
%	Modulus
?:	Conditional Operator
Comparison Operators	
==	Equality
!=	Not Equal
<	Less Than
>	Greater Than
<=	Less Than or equal to
>=	Greater than or equal to

Type Construction Operators

Operator	**Description**	**Example**
()	Tuple constructor operator − This operator constructs a tuple.	(Raja, 30)
[]	Map constructor operator − This operator construct a tuple.	[name#Raja, age#30]
{}	Bag constructor operator − To construct a bag, we use this operator.	{(Raja, 30), (Mohammad, 45)}

Loading and Storing

LOAD	It loads the data from a file system into a relation. Example: grunt>student = LOAD 'hdfs://localhost:9000/pig_data/student_data.txt' USING PigStorage(',') `as ( id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray );`

STORE	It stores a relation to the file system (local/HDFS). Example: grunt> STORE student INTO ' hdfs://localhost:9000/pig_Output/ ' USING PigStorage (',');

Filtering

FILTER	There is a removal of unwanted rows from a relation.
DISTINCT	We can remove duplicate rows from a relation by this operator.
FOREACH, GENERATE	It transforms the data based on the columns of data. Example :`grunt> Relation_name2 = FOREACH Relatin_name1 GENERATE (required data);` `grunt> foreach_data = FOREACH student_details GENERATE id,age,city;`
STREAM	To transform a relation using an external program.

FOREACH-GENERATE

Assume that we have a file named student_details.txt in the HDFS directory /pig_data/ as shown below.
student_details.txt

```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
```

Load this file into Pig with the relation name student_details as shown below.

```
grunt> student_details = LOAD
'hdfs://localhost:9000/pig_data/student_details.txt' USING PigStorage(',')
   as (id:int, firstname:chararray, lastname:chararray,age:int,
phone:chararray,city:chararray);
```

Get the id, age, and city values of each student from the relation student_details and store it into another relation named foreach_data using the foreach operator

```
grunt> foreach_data = FOREACH student_details GENERATE id,age,city;
```

Verification

Verify the relation **foreach_data** using the **DUMP** operator as shown below.

```
grunt> Dump foreach_data;
```

Output

It will produce the following output, displaying the contents of the relation **foreach_data**.

```
(1,21,Hyderabad)
(2,22,Kolkata)
(3,22,Delhi)
(4,21,Pune)
(5,23,Bhuwaneshwar)
(6,23,Chennai)
(7,24,trivendram)
(8,24,Chennai)
```

Grouping and Joining

JOIN	We can join two or more relations.
COGROUP	There is a grouping of the data into two or more relations.
GROUP	It groups the data in a single relation.
CROSS	We can create the cross product of two or more relations.

Sorting

ORDER	It arranges a relation in an order based on one or more fields.
LIMIT	We can get a particular number of tuples from a relation.

Combining and Splitting

UNION	We can combine two or more relations into one relation.
SPLIT	To split a single relation into more relations.

Diagnostic Operators

DUMP	It prints the content of a relationship through the console.
DESCRIBE	It describes the schema of a relation.
EXPLAIN	We can view the logical, physical execution plans to evaluate a relation.
ILLUSTRATE	It displays all the execution steps as the series of statements.