**Apache Pig:**

* Apache Pig is a high-level platform for processing large datasets, representing Big Data as data flows.
* It provides a high-level scripting language, known as *Pig Latin* which is used to develop the data analysis codes.
* First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language.
* Internally *Pig Engine* converted all these scripts into a specific map and reduce task, abstracting the complexity from programmers.
* Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

**Apache Pig – Architecture:**

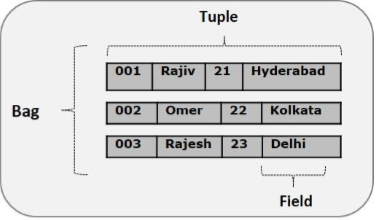


**Apache Pig Components:**

1. **Parser**: Initially the Pig Scripts are handled by the Parser. Checks the Pig script for errors and turns it into a flowchart (DAG) showing the steps and data flow.
2. **Optimizer**: Improves the flowchart by removing unnecessary steps to make it more efficient.
3. **Compiler**: Converts the optimized flowchart into MapReduce jobs.
4. **Execution Engine**: Submits the compiled MapReduce jobs to Hadoop for execution, producing the desired results.

**Pig Data Model :**

The data model of Pig Latin is fully nested and it allows complex non-atomic datatypes such as **map** and **tuple**.



**Atom**: It is a atomic data value which is used to store as a string. The main use of this model is that it can be used as a number and as well as a string.

**Example** − ‘raja’ or ‘30’

**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.

**Example** − (Raja, 30)

**Bag:** A collection of tuples is known as a bag , allowing for flexible schemas where tuples may have varying numbers or types of fields. A bag is represented by ‘{}’. A bag is similar to a table in a database but does not enforce uniformity in its structure.

**Example** − {(Raja, 30), (Mohammad, 45)}

**Map:** A collection of key-value pairs where keys are unique strings, and values can be of any type. Maps are represented using [].

**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).

example: {(Raja, 30), (Mohammad, 45), (John, 25)}

**Pig Latin:**

**Pig Latin Statements:**

The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.

* It can span multiple lines.
* Each statement must end with a semi-colon.
* It may include expression and schemas.

**Pig Latin – Data types:**

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: 1970-01-01 T 00:00:00.0

**Developing and testing Pig Latin scripts:**

**1. Set Up**

* Install **Hadoop** (for big data) and **Pig** (for writing scripts).
* Start **Hadoop** and **Pig** on your computer.

**2. Write Your Script**

A Pig script processes data.

-- Load data from a file

data = LOAD 'input.txt' USING PigStorage(',') AS (name:chararray, age:int);

-- Keep only people who are 18 or older

adults = FILTER data BY age >= 18;

-- Group people by age

grouped\_data = GROUP adults BY age;

-- Count how many people are in each group

age\_counts = FOREACH grouped\_data GENERATE group, COUNT(adults);

-- Save the results

STORE age\_counts INTO 'output.txt' USING PigStorage(',');

**3. Run the Script**

* **Local Mode** (for small data):
* pig -x local script.pig
* **MapReduce Mode** (for large data):
* pig script.pig

**4. Test and Check**

* Use DUMP to print data to the screen for testing:
* DUMP age\_counts;
* Use EXPLAIN to see how the script will run:
* EXPLAIN data;

**5. Make it Faster**

* Use **LIMIT** for smaller tests.
* Run jobs in **parallel** for large datasets:
* STORE age\_counts INTO 'output' USING PigStorage(',') PARALLEL 5;

**6. Store Results**

After running the script, save the results to a file.

Example: STORE age\_counts INTO 'output\_dir' USING PigStorage(',');

**Advantages:**

1. **Easy to Use:** Simplifies complex data processing.
2. **Flexible:** Can handle various data operations.
3. **Customizable:** Allows creating custom functions.
4. **Optimized:** Automatically improves performance.

**Disadvantages:**

1. **Learning Required:** Users need to learn Pig Latin.
2. **Speed:** May be slower than custom MapReduce code.
3. **Debugging:** Can be tricky to debug.
4. **Resource Use:** Can use a lot of resources for big tasks.

**Applications:**

1. **Data Processing:** Used for processing large datasets in various industries.
2. **ETL (Extract, Transform, Load):** Commonly used for data extraction, transformation, and loading tasks.
3. **Data Analysis:** Helps in analyzing large volumes of data for insights and trends.
4. **Prototyping:** Useful for quickly prototyping data processing applications.