Apache PIG Architecture

Table of Contents

Overview

- Abstraction over MapReduce
- Analyze data sets as data flows
    - In Pig, data processing tasks are described as a sequence of operations (transformations) that data passes through. These operations (such as filtering, joining, grouping, or aggregating data) are applied one after another, resembling a flow of data from one stage to the next.
- Language used is Pig Latin, which is a high level language
- Pig Engine Pig Latin Scripts as input and converts the scripts into MapReduce
- Developed by Yahoo
- Easier than MapReduce

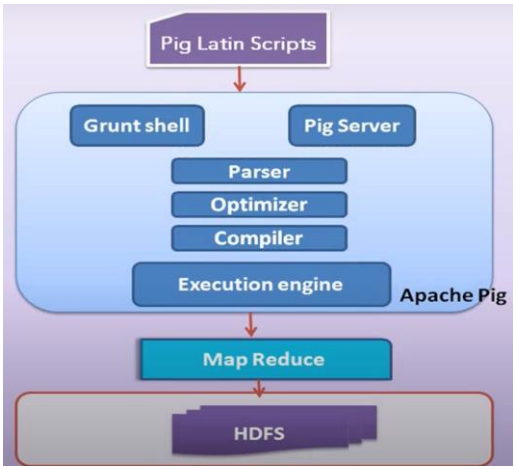
Features of Pig

- Rich Operations Sets: Joins, sort, group, filter, etc.
- Ease of Programming
- Self-Optimization: Converts into MR Jobs
- UDF's: UFD's in Java, and invoke in Pig Scripts
- Handles all kinds of data: Structured, Unstructured (with help of UDF and other tools), and Semi-Structured.
- Stores results in HDFS.

Pig Vs MapReduce

	Apache Pig	MapReduce
1	Apache Pig is a data flow language.	MapReduce is a programming style
2	It is a high level language English like language	MapReduce is usually a compiled java code
3	Syntax for performing join on multiple files is very intuitive, simple and SQL like	MapReduce code can become quite complex for joining files
4	Learning curve in Apache Pig is very small	Expertise in Java and the MapReduce libraries is a must
5	Apache Pig scripts can do the equivalent of multiple lines of MapReduce code	Generally MapReduce takes more lines of code to perform the same operations
6	Debugging, testing Pig scripts is very easy	Like any java program, MapReduce programs take time for code testing, unit testing etc

Pig Architecture



- Programmers write commands in Pig Latin
- A script file contains a number of Pig Latin statements.
- Then the script is converted into series of MR Jobs, and executed on HDFS files.
- **Parser:**
    - Checks the syntax of the script
    - Does Type checking
    - Output of Parser is a DAG (Directed Acyclic Graph)
    - **DAG:**
      Consider a simple script that performs the following tasks:
      1. Load a dataset.
      2. Filter the dataset to only include records from 2023.
      3. Group the filtered data by customer ID.
      4. Calculate the total transaction amount for each customer.

      The corresponding DAG will be :
      [Load Data] --> [Filter Data] --> [Group by Customer ID] --> [Calculate Total Amount]

      The benefit is that Pig can know which is a serial, and which task can be executed in parallel independently.
- **Optimizer:**
    - Caries out Logical optimizations such as projection, and pushdown.
        - **Projection:**
          Selecting only the required columns from the dataset instead of retrieving all columns.
        - **Pushdown:**
            - Refers to the practice of moving data filtering or transformation operations closer to the source of the data (e.g., the database or file system) before transferring the data for further processing.

- If you have a dataset with 10 columns but only need 2 of them, or if you have millions of records but only need a subset of them (say, those where `status = 'active'`), pushdown would attempt to apply these filters or projections at the source.
- By doing so, only the relevant data is brought forward, reducing the workload for later stages.

- **Compiler:**
  - Compiles the optimized logical plan into a series of MR Jobs.
- **Execution Engine:**
  - Submits the MR Jobs to Hadoop

Pig Execution Option

1. Interactive/Grunt Mode on Local File System
   a. Command "Pig –x local"
   b. Runs in a single machine (local)
   c. All files are in the local file system
2. Interactive/Grunt Mode in MR or HDFS
   a. Command "Pig"
   b. Runs on Hadoop Cluster
   c. Is the default mode
3. Batch Mode or Script Mode
   a. Command "pig scriptname.pig"
   b. This script is a file containing all the Pig Latin Commands
4. Embedded Mode (UDF)
   a. Runs UDF based on programming languages such as Java
   b. These UDFs are used in the scripts and can be invoked through pig command for scripts.

Pig Data Types

1. **Atom**
   a. Integers
   b. Float
   c. Doubles
   d. CharArray
   e. ByteArray (Default datatype, if the datatype of a variable is not declared)
2. **Tuples**
   a. Formed by grouping atoms
   b. E.g.: (Abhishek, 1.58, 6)
3. **Bag**
   a. Combination of Tuples
   b. {('Abhishek',5),('Rajesh', 2.5)}
4. **Map**
   a. Key Value Pairs
   b. ['name'#'Abhishek','Height'#5.7]

Creating an example script

```
employee = Load 'people' as (empid,name,hours) ;
parttime = FILTER employee BY Hours < 20 ;
sorted = ORDER parttime BY hours DESC ;
STORE sorted INTO 'part_time' ;
DUMP sorted ; (To display contents of bag on screen)
DESCRIBE sorted ; (To view the structure of the bag)
```