Pig

What is Pig?

- Apache Pig is a platform for data analysis.
- It is an alternative to MapReduce programming.
- It was developed at Yahoo.
- It is widely used for ETL - Extract, Transform and Load.
    - Pig can extract data from various sources, apply various various operations on the data and load the data into data warehouse.

Anatomy (components) of Pig

Pig has four main parts :

- Pig Latin
  - Data flow language of pig.
- Grunt shell
  - Interactive shell where you can type Pig Latin Statements.
- Pig interpreter
  - Processes and parses Pig Latin Statements
  - Checks data types.
  - Performs optimization.
- Execution engine
  - Creates MapReduce jobs and submit it to hadoop.
  - Monitors progress of jobs.

Pig supports :

- HDFS commands
- UNIX shell commands
- Relational operators
- Positional parameters
- Common mathematical functions
- User Defined Functions
- Complex Data Structures

Pig Philosophy :

- Pig eats anything
  - Pig can process both structured and unstructured data.
- Pig lives anywhere
  - Pig can process file both on HDFS and local file system.
- Pig is domestic animal
  - In Pig you can develop user defined functions.
- Pig fly
  - Pig can process data quickly.

Pig program flow :

- **LOAD** statement that reads data from file system.
- Series of statements to perform transformations.
- **DUMP** or **STORE** to display or store result.

Pig Latin overview:

- Keywords
  - These are reserved words
  - Example - load , filter , foreach , into , store , dump
- Identifiers
  - Names given to variables or relations
  - Identifier should start with an alphabet and can contain alphabet, numbers and underscore
- Comments
  - Single line comment start with “--”
  - Multiline comments start with “/*” and end with “*/”
- Case sensitive
  - Keywords are not case sensitive
  - Relations, paths and function names are case sensitive

Operators in Pig

- Arithmetic
  - +, -, *, /, %
- Comparison
  - ==, !=, >, <, >=, <=
- Null
  - Is null, is not null
- Boolean
  - And, or, not

Data types in Pig

- Simple data types
  - Int
  - Long
  - Float
  - Double
  - Chararray
  - Bytearray
  - Datetime
  - Boolean
- Complex data types
  - Tuple - ordered set of elements
  - Bag - collection of tuples
  - Map - key, value pair

Modes of pig :

- Interactive mode
  - Grunt shell
- Batch mode
  - Statements are saved in file ending with .pig extension

Execution modes of pig :

- Local mode
  - pig -x local filename
  - This runs pig on local file system
- MapReduce mode
  - You need to have access to hadoop cluster to read/write file
  - pig filename
  - This is default mode of pig

Relational operators

1. Filter
    a. Select tuples from a relation based on some condition
    b. B = filter A by gpa > 4;
2. Foreach
    a. To iterate over each tuple of the relation
    b. B = foreach A generate UPPER (name), gpa
3. Group
    a. group data based on column in a tuple
    b. B = group A by gpa;
4. Distinct
    a. Used to remove duplicate tuples
    b. B = distinct A;
5. Limit
    a. Limit the number of output tuples
    b. B = limit A 3;
1. Order by
    a. To sort a relation based on specific column/value
    b. B = order A by name;
2. Join
    a. Join two or more relations based on values in the common field
    b. C = join A by rollno, B by rollno;
3. Union
    a. Merge the content of two relations
    b. C = union A, B;
4. Split
    a. Partition a relation into two or more relations
    b. split A into X if gpa == 4, Y if gpa <4;
5. Sample
    a. Select random sample of data based on sample size
    b. B = sample A 0.01;

Eval functions

1. AVG
    a. B = group A by name;
    b. C = foreach B generate A.name, AVG(A.marks);
2. MAX
    a. B = group A by name;
    b. C = foreach B generate A.name, MAX(A.marks);
3. MIN
    a. B = group A by name;
    b. C = foreach B generate A.name, MIN(A.marks);
4. SUM
    a. B = group A by name;
    b. C = foreach B generate A.name, SUM(A.marks);
5. COUNT
    a. B = group A by name;
    b. C = foreach B generate A.name, COUNT(A);

Complex data types

- Tuple
  - Input : (john,12) (james,7)
  - A = load “filename” as (t1:tuple(t1a:chararray, t1b:int), t2:tuple(t2a:chararray,t2b:int));
- Map
  - Input : john [city#banglore]
  - A = load “filename” using PigStorage as (name:chararray,m:map[chararray]);
  - B = foreach A generate m#’city’ as cityname:charrray;

Word count using pig

```
Lines = LOAD ‘file.txt’ as (line:chararray);

Words = foreach lines GENERATE FLATTEN(TOKENIZE(line)) as word;

Grouped = GROUP words BY word;

Wordcount = FOREACH grouped GENERATE group, COUNT(words);

DUMP wordcount;
```