# APACHE PIG

JPA Solutions
Center Of Excellence

# Introduction

- Abstraction over Mapreduce
- It is a data-flow language called Pig Latin
- Pig was originally created at Yahoo! To serve the similar need to hive.
- Many developers doesn't have the knowledge of Java/Mapreduce
- Under the covers,  PigLatin scripts are turned as a Mapreduce jobs and runs on the hadoop cluster

## Usecases

- Data factory usecase – ETL
  - Most of the mainframe jobs are converted into Pig Based Jobs
- Rapid prototyping of algorithms for processing large dataset
- Adhoc queries across large data sets
- Data processing for web search platforms
- Web log processing

## PIG Features

- Joining the dataset
- Sorting and aggregation
- Grouping data
- Referring to elements by position(useful for large datasets)
- creation of UDF using java

## Installation

- tar -xvf pig-***.tgz
- Set JAVA_HOME
- Set HADOOP_HOME
  - Instead you can set properties in pig.properties

## Accessing PIG

- Interactive mode
  - Grunt, the Pig shell
- Batch mode
  - Submitting a Pig script directly
- Pig server
  - Java class, JDBC like interface

## First Script – Grunt (bin/pig)

```
= load '/datagen_10.txt' using PigStorage(',') ;
F = filter A by $2 == 'avil';
dump F;
```

## Alias name to the fields with datatypes

- A = load '/user/senthil/drugdata' using PigStorage(',') as (**pid:int, pname:chararray, drug:chararray, gender:chararray, tot_amt:int**);
- F = filter A by drug == 'avil';
- dump F;

# Data Types

- Scalar Types
  - Int 10
  - float 10.0F
  - long 10L
  - double 10.0
  - chararray hello
  - bytearray
- Complex Types
  - Map [key#value]
  - Tuple(100,senthil)
  - Bag((100,senthil),(100))
- Null

## Data Formats

- PigStorage
  - using field delimited text format
- BinStorage
  - Loads/stores relations in HDFS from or to binary files
- BinaryStorage
  - Loads/stores relations in HDFS containing only a single field tuples with a value of bytearray
- TextLoader
  - Loads relations in HDFS from a plain text format
  - Loads a whole line as single column
- PigDump
  - Stores relations in HDFS by writing the toString() representation of tuples, one per line

## Store the results

- = load '/datagen_10.txt' using PigStorage(',') ;
- F = filter A by $2 == 'avil';
- Store F  in '/pig_result001' using PigStorage(',') ;

## Viewing the Schema

- A = load '/user/senthil/drugdata' using PigStorage(',') ;
- F = filter A by $2 == 'avil';
- **Describe F;**
- **Describe  A;**

## Execution Plan

- A = load '/user/senthil/drugdata' using PigStorage(',') ;
- F = filter A by $2 == 'avil';
- **Explain F;**

## Grouping & Sorting

- A =load '/user/senthil/drugdata' using PigStorage(',');
- D = GROUP A by $2;
- sm = foreach D generate group,SUM(A.$4) as s;
- smorder = order sm by s desc;
- dump smorder;

## Eliminating duplicates

- **Select distinct drug from patient;**
  - A = load '/user/senthil/drugdata' using PigStorage(',') as (pid:int, pname:chararray, drug:chararray,gender:chararray,tot_amt:int);
  - D = foreach A generate drug;
  - unique = DISTINCT D;
  - Dump unique;

## LIMIT , match and non-match

- **-- LIMIT - Reduce the number of o/p records**
  - A = load '/user/senthil/drugdata' using PigStorage(',') as (pid:int, pname:chararray, drug:chararray,gender:chararray,tot_amt:int);
  - F = limit A 2;
  - dump F;

- **--Similar to Like in SQL**
  - A = load '/user/senthil/drugdata' using PigStorage(',') as (pid:int, pname:chararray, drug:chararray,gender:chararray,tot_amt:int);
  - F = filter A by pname matches 'Brandon.*';
  - dump F;

## Contd..

- **-- Not matches Brandon**
  - A = load '/user/senthil/drugdata' using PigStorage(',') as (pid:int, pname:chararray, drug:chararray,gender:chararray,tot_amt:int);
  - F = filter A by not pname matches 'Brandon.*';
  - dump F;

## Contd..

- A =load '/user/senthil/drugdata' using PigStorage(',');
- F = GROUP A ALL;
- sm = foreach F generate COUNT_STAR(A);
- dump sm;

## Macros in Pig

- DEFINE my_macro(V, col,value) returns B {
  $B = FILTER $V BY $col == '$value';
  };
- A = load ‘/datagen_10.txt' using PigStorage(',');
- C = my_macro(A,$2,'metacin');
- dump C;

## Joining DataSets

- PigLatin supports inner and outer joins of two or more relations.

Inner join – Join two tables by common key

- = load '/datagen_10.txt' using PigStorage(',');
- B=load `/drug.txt' using PigStorage();
- C=join A by $2, B by $0;
- dump C;

## Outer joins

- Pig can perform left, right, full outer joins(similar to sql)

- =load ‘/datagen_10.txt' using PigStorage(',');
- B = load '/drug.txt' using PigStorage();
- C = join A by $2 [left outer|right outer|full outer], B by $0;
- Dump C;

## Special Joins

- Replicated Join  or (MapSide Join)
- Merge Join
- Skewed Join

## Group Vs CoGroup

- GROUP - collects records of one input based on a key
- COGROUP - collects records of n inputs based on a key
- C = COGROUP A by $2, B by $0;
- Dump C;

## SPLIT

- Partition a relation into two or more relation
- A = load '/user/senthil/drugdata' using PigStorage(',') as (pid:int, pname:chararray, drug:chararray, gender:chararray,tot_amt:int);
- SPLIT A into males IF gender == 'male', females IF gender =='male';

## Pig Scripts

- Use Pig scripts to place Pig Latin statements and Pig commands in a single file.
- Good practice to identify the file using *.Pig
- Can run scripts that are stored in HDFS
- Pig hdfs://path/script.pig

Single as well as Comment lines can be added

# Pig Server

- It is not a daemon server
- It is a single threaded stub to run pig in a java application
  - org.apache.pig.Pigserver class
- Allows java programs to invoke pig commands
- Use “local” or “mapreduce” to indicate run method
- PigServer
  - ps = new PigSrever(“local”)
  - ps.registerQuery(“        = load 'file' ”)
  - ps.registerQuery(“B = group    by $0 ”)
  - ps.store(“B”, “outfile”)

# Case-sensitivity

- Case-sensitive
- Keywords - (load,using,filter,ls, etc)
- Case-insensitive
- Aliases(A,B),functions(COUNT, AVG,etc)

## Implementation of UPPER UDF

```
package com;
public class Upper extends EvalFunc<String> {
    @Override
public String exec(Tuple input) throws IOException {
    if (input == null || input.size() == 0) {
return null;}
try {String str = (String) input.get(0);
return str.toUpperCase();
} catch (IOException e) {
    e.getMessage();}
return null;}}
```

# THANK YOU!!