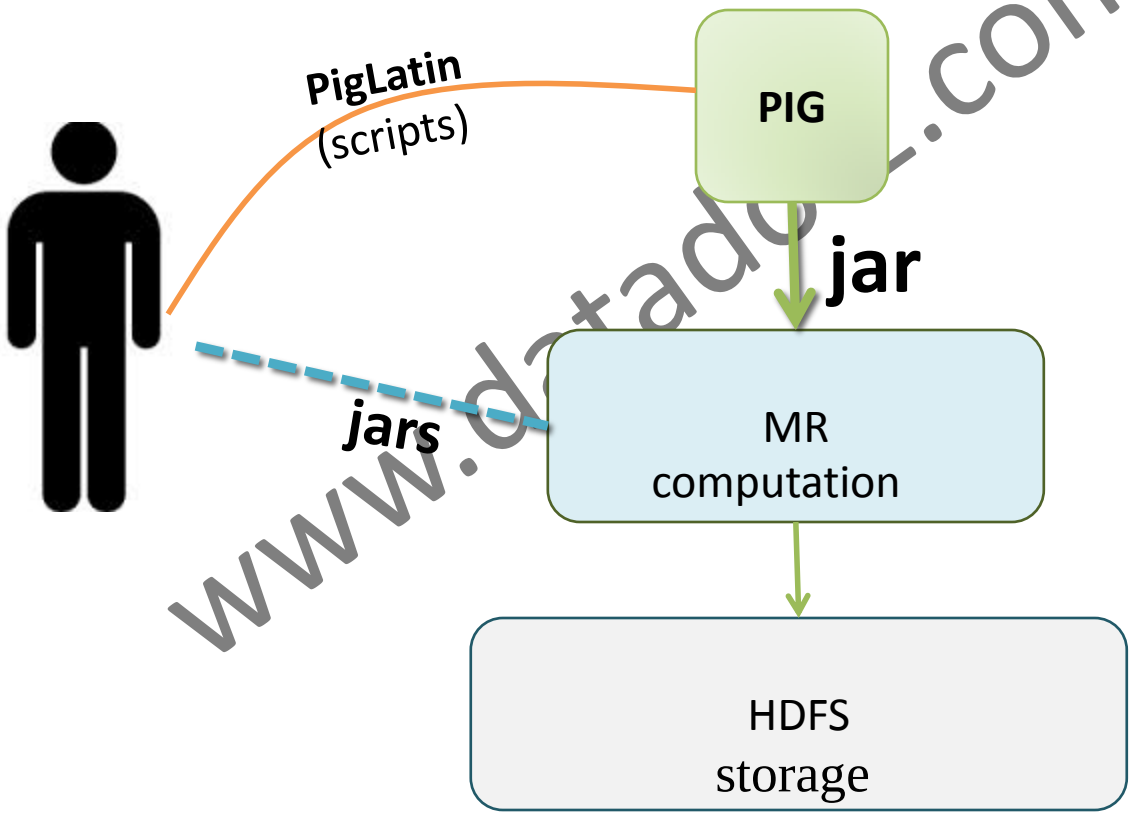
APACHE PIG

Senthil Kumar A

Pig – Abstraction Component for MapReduce



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

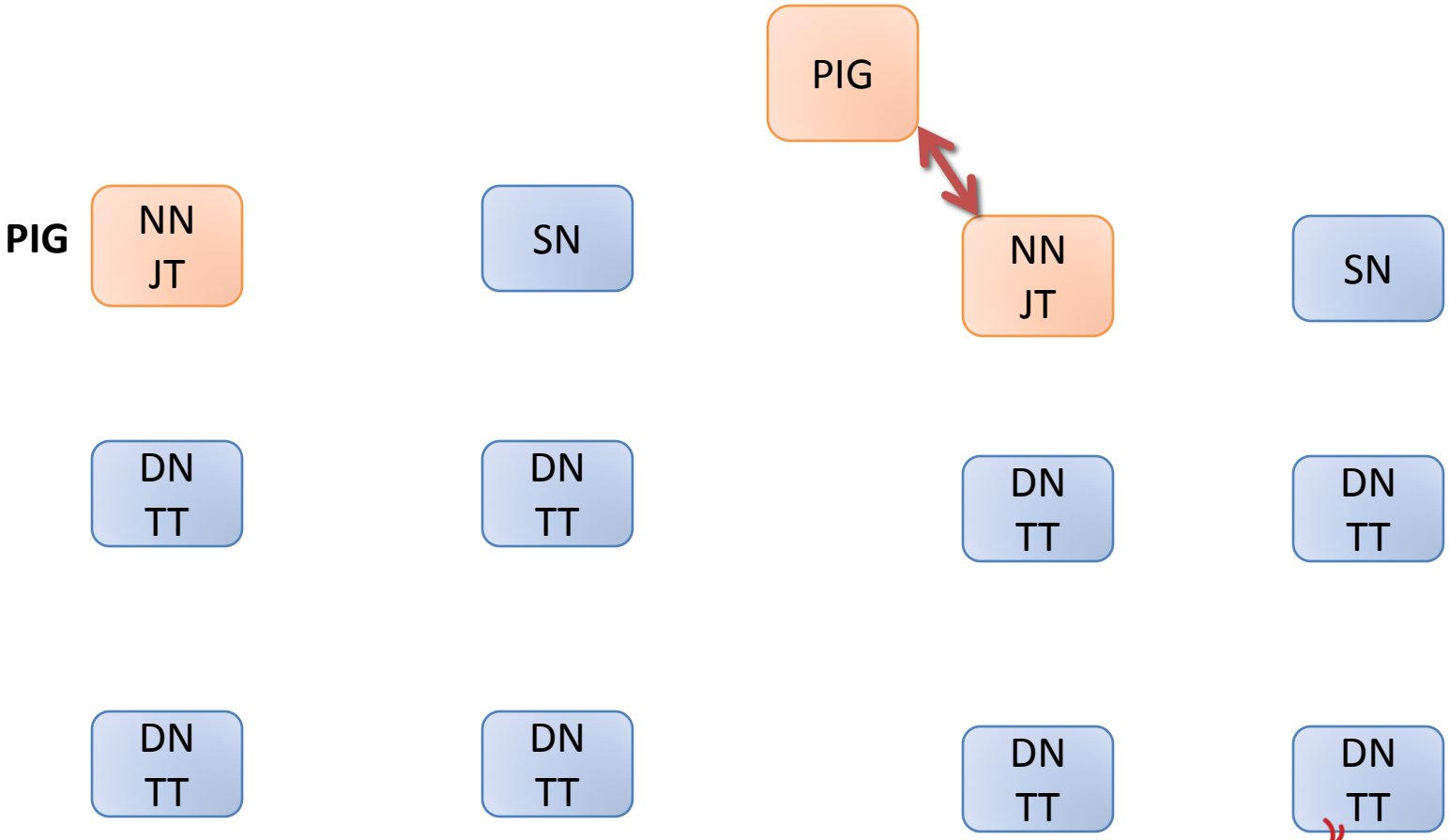
Pig Features

- 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

Installation in Production Cluster



Install inside cluster

Install outside cluster

Hive	Pig
HiveQL (SubSet of SQL-92)	Grunt (dataFlow)
Declarative	Procedural
Schema Defined	No . But can be optionally Defined runtime
Table	Bag (analogous but immutable)
Record	Tuple
Columns	Fields
Connectivity – JDBC	PigServer

Pig Execution Modes

- **Mapreduce Mode**
  - bin/pig (Mapreduce is the default) **OR**
  - bin/pig -x mapreduce
  - Runs the jobs in the hadoop cluster and reads/writes to HDFS
- **Local Mode**
  - bin/pig -x local
  - (Good for debugging on local data sets)
  - Uses Hadoop's LocalJobRunner and the local file system

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)

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

*select * from patient where drug = 'avil';*

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 (Knowledge)

- **Scalar Types**
  - int 10
  - float 10.0F
  - long 10L
  - double 10.0
  - chararray hello
  - bytearray

Data formats

- PigStorage
  - using field delimited text format
- BinStorage
  - Loads/stores relations in HDFS from or to binary files
- 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

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

Store -> writes the data in HDFS directory

Viewing the Schema

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

- **Illustrate A;**
- **Illustrate F;**

grouping and sorting

select drug, sum(amt) as tot from patient group by drug order by tot

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

group-Implicit field name given to the group key

- 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 ..

- **select count(*) from patient;**
- 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

- A =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)

- A =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;

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;

- Note : Check difference between cogroup and full outer join

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

- bin/pig –x local myscript.pig
- bin/pig –x mapreduce myscript.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(“A = load 'file' ”);
  - ps.registerQuery(“B = group A by $0 ”);
  - ps.store(“B”, “outfile”);

Register

- By specify the Pig script path where It resides
- Usage: Register /home/user/myscript.pig
- Execution
- Using the commands run/exec
- Usage:
- exec myscript.pig (Batch mode)
- run myscript.pig (Interactive mode)

UDF

- Write a java class that extends EvalFunc and implements the exec method compile and package into jarTell pig about the jar
- using the REGISTER keyword
- Public class MyFunc extends EvalFunc {
- Public double exec(Tuple input) {
- ....}}
- REGISTER jarname.jar
- DEFINE myFunc com.examples.MyFunc();

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

DATA DOTZ