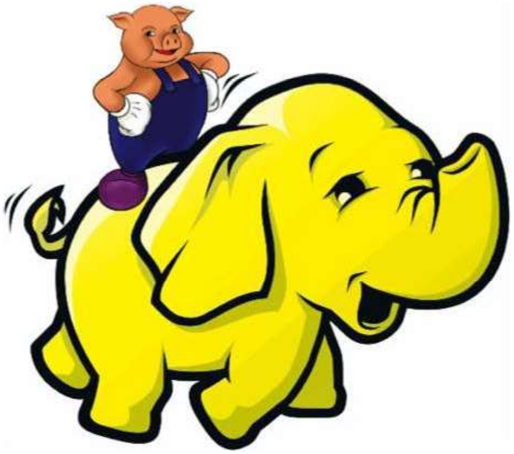


Apache Pig

Extract Transform Load

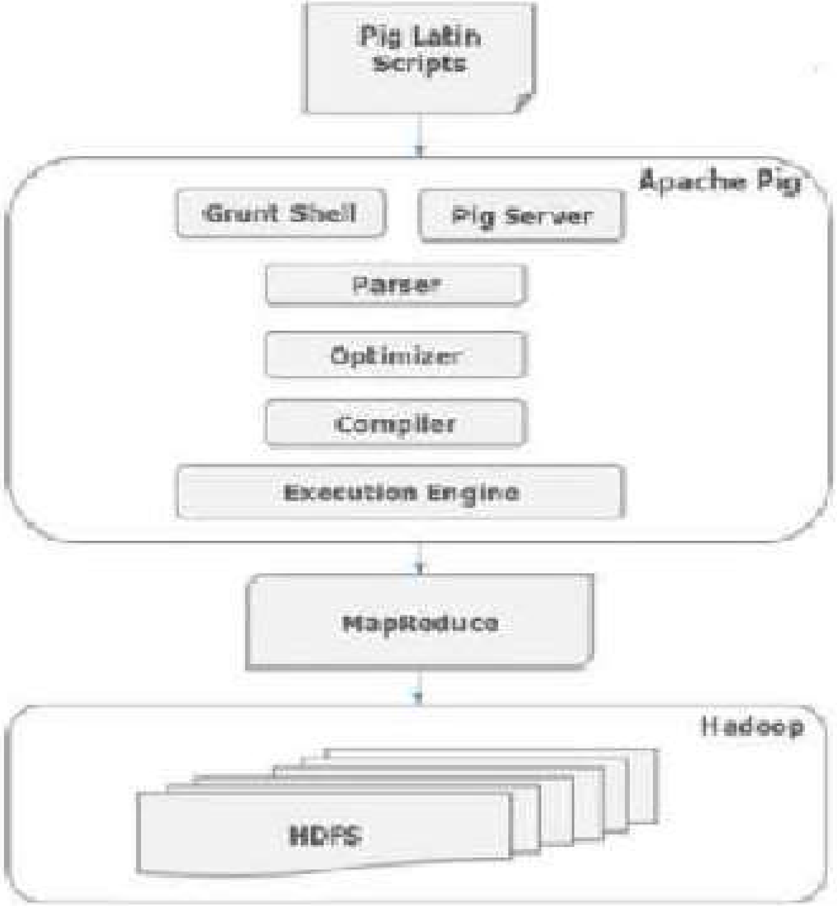
Dineshkumar S
SmartR.in

Introduction

Apache Pig:

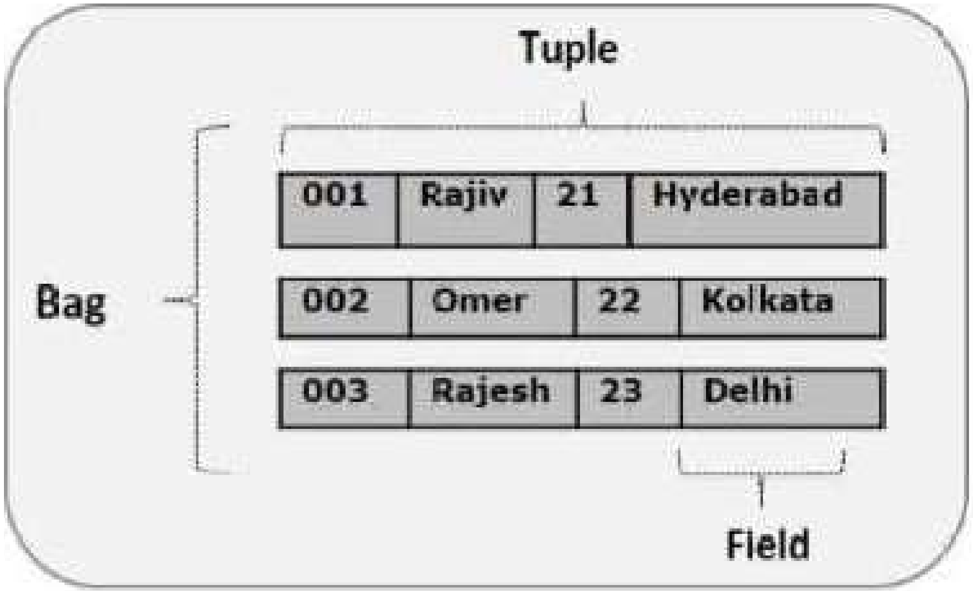
- is an abstraction over MapReduce
- has structure which is amenable for substantial parallelization
- is generally used with Hadoop
- does lazy evaluation
- uses Pig Latin, which provides
    - Ease of Programming
    - Optimization opportunities
        - Focus on semantics rather than efficiency
    - Extensibility
        - Can create UDFs

Architecture



Pig Latin Data Model

- Fully nested & complex data model
- Atom
- Tuple
- Bag
- Map
- Relation



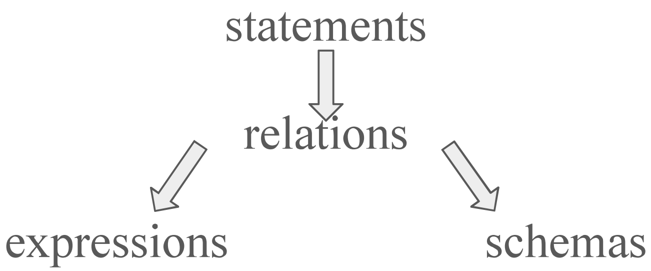
001	Rajiv	21	Hyderabad
002	Omer	22	Kolkata
003	Rajesh	23	Delhi

Pig Environment

Establishing Pig environment contains following three major steps:

- Installation
  - https://pig.apache.org/docs/r0.7.0/setup.html
- Grunt shell
  - Pig interactive shell which accepts Pig-latin based statements.
- Execution
  - Two modes of execution, Mapreduce mode and Local mode, we’ll see both ahead.

Pig Latin



statements

relations

expressions

schemas

Data Types

Basic	Complex
int, long, biginteger	Tuple : (raju, 5)
float, double, bigdecimal	Bag: {(raju,5),(ranu,6)}
chararray	Map: [‘name’#’raju’,’age’:40]
bytearray	
boolean	
datetime	

Operators and Constructs

- All standard Arithmetic and Relational operator
- Constructs
  - Tuple: **()**
  - Bag: **{}**
  - Map: **[]**
- Relational Operators
  - Load & Storing: **LOAD & STORE**
  - Filtering: **FILTER, DISTINCT, FOREACH, GENERATE, STREAM**
  - Grouping & Joining: **JOIN, COGROUP, CROSS, GROUP**
  - Sorting: **ORDER, LIMIT**
  - Combining & Splitting: **UNION, SPLIT**
  - Diagnostic & Debugging: **DUMP, DESCRIBE, EXPLAIN, ILLUSTRATE**

Sample Queries

```
student = LOAD 'hdfs://localhost:9000/pig_data/student_data.txt'
  USING PigStorage(',')
  as ( id:int, firstname:chararray, lastname:chararray, phone:chararray,
  city:chararray );
```

```
STORE student INTO ' hdfs://localhost:9000/pig_Output/ ' USING PigStorage (',');
```

```
Dump student
```

```
describe student;
```

```
explain Relation_name;
```

```
illustrate student
```

```
group_data = GROUP student_details by age
```

```
cogroup_data = COGROUP student_details by age, employee_details by age;
```

```
result = JOIN relation1 BY columnname, relation2 BY columnname;
```

Sample Queries (contd.)

```
cross_data = CROSS customers, orders;

student = UNION student1, student2;

SPLIT student_details into student_details1 if age<23, student_details2 if (22<age and age>25);

filter_data = FILTER student_details BY city == 'Chennai';

distinct_data = DISTINCT student_details;

foreach_data = FOREACH student_details GENERATE id,age,city;

order_by_data = ORDER student_details BY age DESC;

limit_data = LIMIT student_details 4;
```

Eval & Load/Store Functions

Eval:

AVG() **BagToString()** **CONCAT()** **COUNT()** **COUNT_STAR()**

DIFF() **IsEmpty()** **MAX()** **MIN()** **SIZE()** **SUM()**

Load/Store:

PigStorage()

TextLoader()

BinStorage()

UDFs(User Defined Functions)

- UDF ?
- Piggybanks?
- Types of UDFs:
  - Filter Functions
  - Eval Functions
  - Algebric Functions
- Writing UDFs
- Registering UDFs
- Defining Alias
- Using UDFs

documentation

UDF’s (contd.)

What is UDF?

In addition to the built-in functions, Pig provides excellent support for User Defined Functions(UDF’s), where we can define our own functions and use them along with different Pig Latin built-in functions. Complete support to write UDF’s is provided in Java and partially in other languages, viz. Jython, Python, JS, Ruby and Groovy.

Piggybanks?

Apache Pig provides a central repository for Java based UDF’s using which you can use the UDF’s written by other people and you can also contribute there.

UDF Example

- This example is written for converting given string to Uppercase,
- You can download the java code from link.
- Extract it and create the jar file from the source code or download from here.
- Now assume we have the jar file at /home/hduser/hadoop/exampleJARs/udfEx1.jar and some sample data at /hduser/sampleData/sampledata.txt
- Now on grunt shell

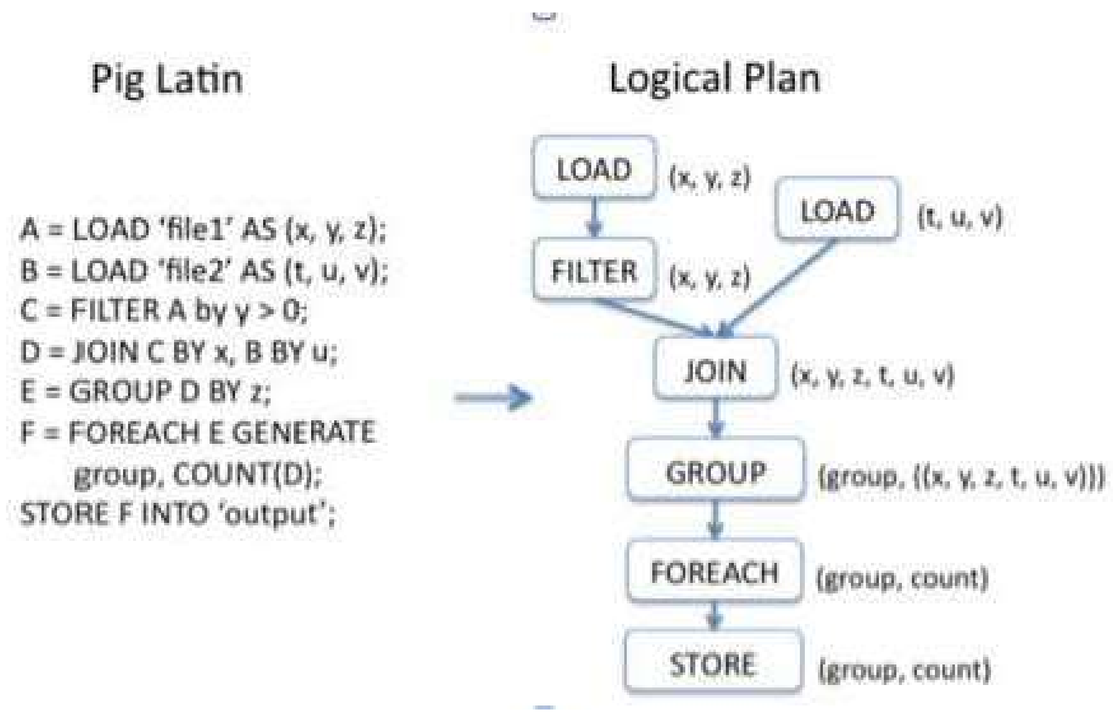
```
REGISTER /home/hduser/hadoop/exampleJARs/udfEx1.jar;

DEFINE udf_eval SampleEval();

data = load '/hduser/sampleData/sampledata.txt' using PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);

upper_case_data = foreach data generate udf_eval(name);

store upper_case_data into 'hdfs://localhost:9000/hduser/outputudf15';
```

Pig Scripts

- Modes of running Pig Scripts
  - Mapreduce mode
  - Local mode (-x parameter)

```
pig -x <mode> <pig_script_path>
```

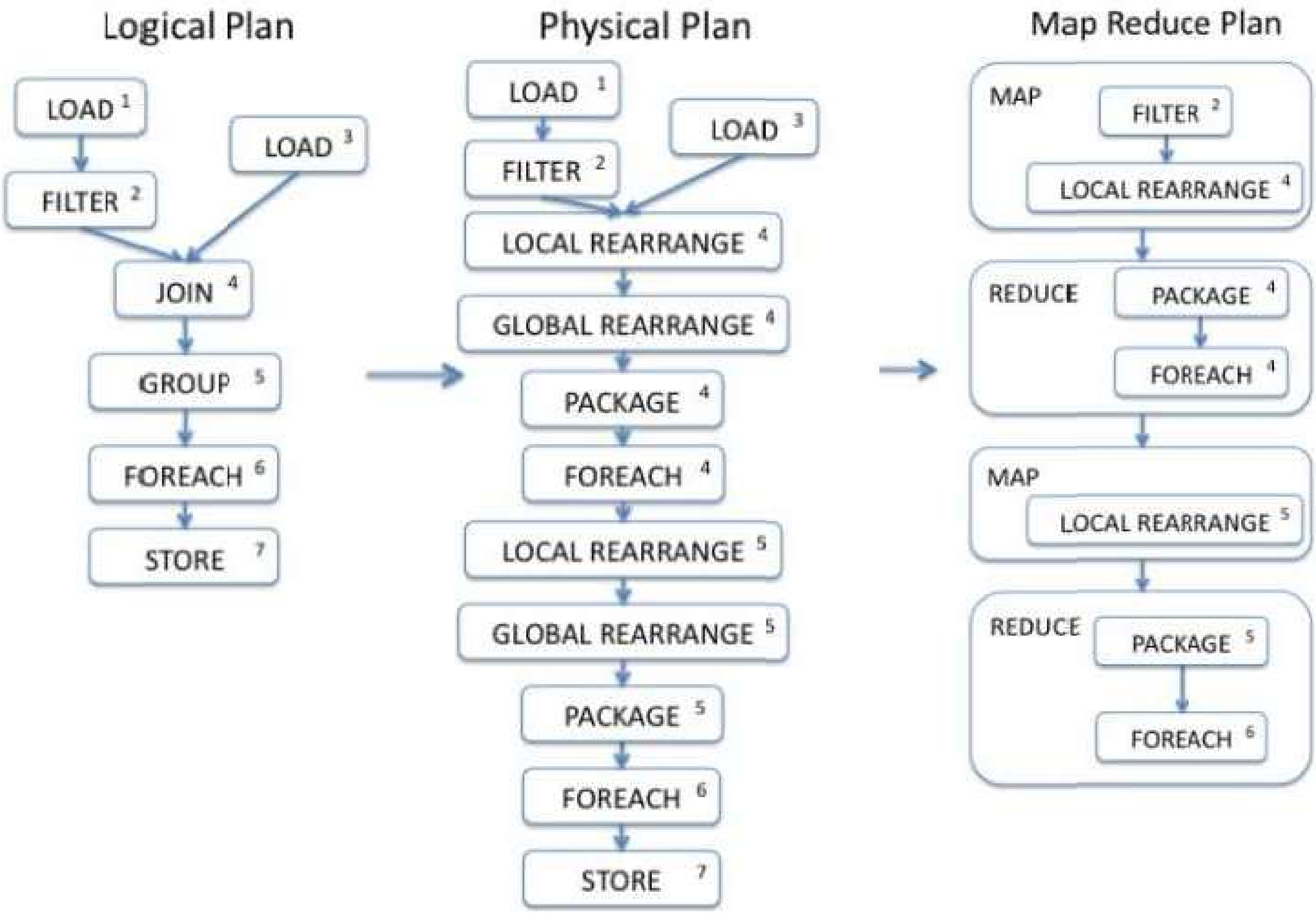
Pig → MapReduce



Pig Latin

```
A = LOAD 'file1' AS (x, y, z);
B = LOAD 'file2' AS (t, u, v);
C = FILTER A by y > 0;
D = JOIN C BY x, B BY u;
E = GROUP D BY z;
F = FOREACH E GENERATE
      group, COUNT(D);
STORE F INTO 'output';
```

Logical Plan



Logical Plan

LOAD 1
LOAD 3
FILTER 2
JOIN 4
GROUP 5
FOREACH 6
STORE 7

Physical Plan

LOAD 1
LOAD 3
FILTER 2
LOCAL REARRANGE 4
GLOBAL REARRANGE 4
PACKAGE 4
FOREACH 4
LOCAL REARRANGE 5
GLOBAL REARRANGE 5
PACKAGE 5
FOREACH 6
STORE 7

Map Reduce Plan

MAP
FILTER 2
LOCAL REARRANGE 4

REDUCE
PACKAGE 4
FOREACH 4

MAP
LOCAL REARRANGE 5

REDUCE
PACKAGE 5
FOREACH 6