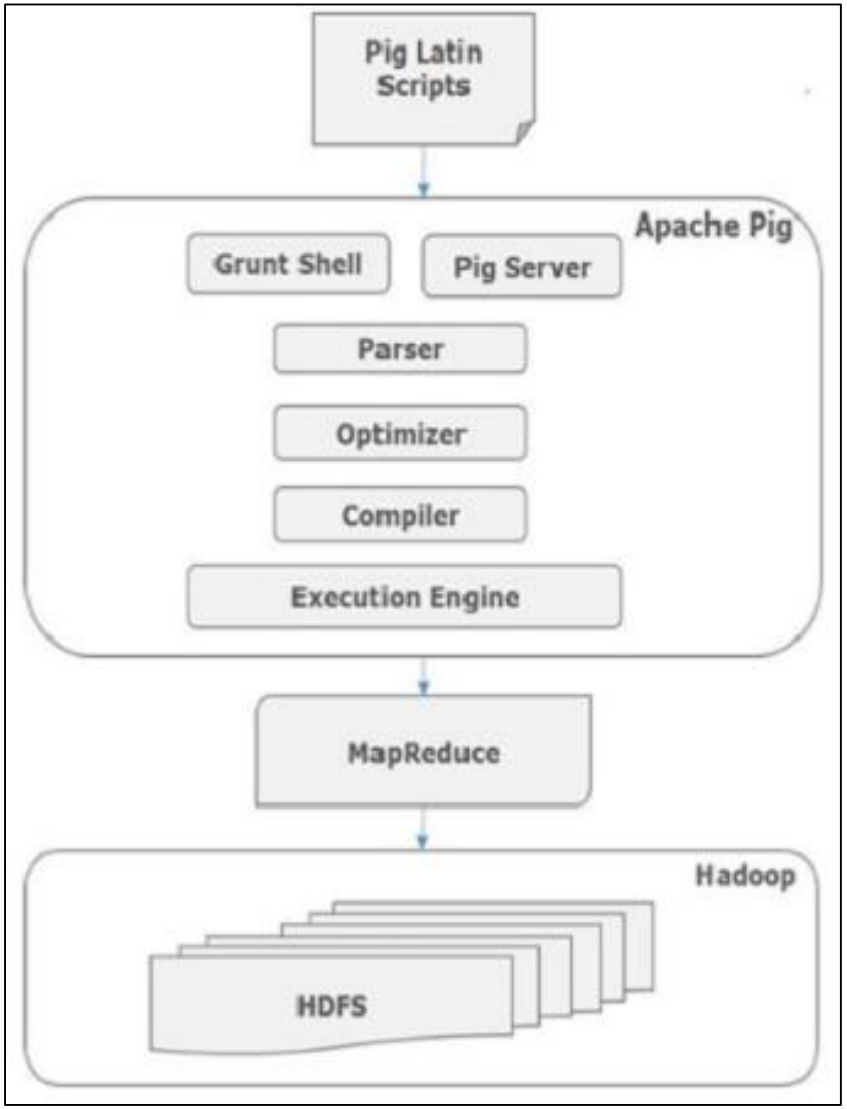
EDU PRISTINE

Apache Pig

Pig

- Introduction to Pig
- Execution Modes & Pig Components [Grunt, Pig Latin, Optimizer, Compiler, Parser]
- Pig Latin Basics
- Case Study
- Developers Alert!!!
- Resources
- Pig vs Hive

PIG Overview



- Pig is a platform for analyzing large data sets
- Pig scripts written in Pig Latin – an easy to learn scripting type of language
- Pig runs on MR as well as Tez – backend engine also
- If Pig using MR, Pig scripts gets converted into optimized MR Jobs

PIG Overview

Definitions

- From the creators: Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs

What Pig Brings to table

- **Ease of programming.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain
- **Optimization opportunities.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency
- **Extensibility.** Users can create their own functions to do special-purpose processing

Pig Overview

So we know,

- current implementation compiles Pig Latin programs into Map/Reduce jobs, and executes them using Hadoop on HDFS file system
- Pig works can work well with semi structured data also. e.g.,: log files
- Pig is a high-level language for writing analytical expressions over such sort of data
- A query planner compiles queries written in this language (called "Pig Latin") into maps and reduces which are then executed on a Hadoop cluster
- Every task which can be achieved using PIG can also be achieved using java implementing in Map reduce

The Pig Power

What Pig Does Best

- Join Datasets
- Sort Datasets
- Filter
- Data Types
- Group By
- User Defined Functions

In Industry its used for

- Mostly processing large amount of raw data Logs , Extract Transform logs (ETL)
- Data Exploration

Pig Components

Pig Latin

- Command based language
- Designed specifically for data transformation and flow expression

Execution Environment

- The environment in which Pig Latin commands are executed
- Currently there is support for Local and Hadoop modes

Pig compiler converts Pig Latin to MapReduce

- Compiler strives to optimize execution
- You automatically get optimization improvements with Pig updates

Pig Execution modes

Local

- Executes in a single JVM
- Works exclusively with local file system
- Great for development, experimentation and prototyping
- Command to Start Pig in Local mode
  - pig –x local

Hadoop Mode

- Also known as MapReduce mode. [This is default mode]
- Pig renders Pig Latin into MapReduce jobs and executes them on the cluster
- Can execute against semi-distributed or fully-distributed hadoop installation
- Command to Start Pig in Hadoop mode

  Simply do not specify any mode, It will start as default mode
  - pig

Pig Configuration

- cat /etc/pig/conf/pig.properties
- Pig.properties file has most of pig configurations which can be changed by root/ admin user of the cluster.
- For example : See exectype = mapreduce. This sets Pig execution engine to Map Reduce. Alternatively You can use any other processing engine for pig like **Tez.**

```
# brief logging (no timestamps)
brief=false

# clustername, name of the hadoop jobtracker. If no port is defined port 50020 will be used.
#cluster

#debug level, INFO is default
debug=INFO

# a file that contains pig script
#file=

# load jarfile, colon separated
#jar=

#verbose print all log messages to screen (default to print only INFO and above to screen)
verbose=false

#exectype local|mapreduce, mapreduce is default
#exectype=mapreduce
# hod realted properties
#ssh.gateway
#hod.expect.root
#hod.expect.uselatest
#hod.command
#hod.config.dir
#hod.param


#Do not spill temp files smaller than this size (bytes)
pig.spill.size.threshold=5000000
#EXPERIMENT: Activate garbage collection when spilling a file bigger than this size (bytes)
#This should help reduce the number of files being spilled.
pig.spill.gc.activation.size=40000000


######################
# Everything below this line is Yahoo specific.  Note that I've made
# (almost) no changes to the lines above to make merging in from Apache
# easier.  Any values I don't want from above I override below.
#
# This file is configured for use with HOD on the production clusters.  If you
# want to run pig with a static cluster you will need to remove everything
# below this line and set the cluster value (above) to the
# hostname and port of your job tracker.

exectype=mapreduce
log.file=
[training@localhost ~]$
```

Pig Execution Mode

Pig Latin

- The programming language used to write Pig queries is called **Pig Latin**

Introduction to

- Grunt

Pig Latin Building Blocks

Building blocks

- **Field -** piece of data
- **Tuple -** ordered set of fields, represented with ( )
- **Bag -** Collection of tuples, represented with “{“ and”}”
- **Map** - Key Value pair represented as key# value

Example

In RDBMS, An employee table contains records. Each record represents an employee. Each employee record describes an employee by employee id, name and age – which are fields

Similarly,

(100, “Joe Smith”, 30) is a Tuple with three fields employee Id, Name and Age.

Employee Table is a Bag as below which has two tuples

{(100, “joe Smith”, 30),(101, “Lee Chong”, “29”)}

Pig Latin Building Blocks [Cont.]

With reference to Relational Database

- Bag is a table in the database
- Tuple is a row in a table
- **Unlike relational table** Bags do not require that all tuples contain the same number of fields *__this means__, {(100, “joe Smith”, 30),(101, “Lee Chong”)} – **Tuples with heterogeneous number of fields is valid** and that’s what make Pig easy to work with semi-structured data.

Generate Complex Data Types

Prepare Input File and put it on HDFS

Tab separated Input File (students.tsv):

joe smith	20	3.5
amy chen	22	3.2
leo allen	18	2.1

1 **Tuple Construction**

```
A = load 'your hdfs path to/students.tsv' as (name:chararray, age:int, gpa:float);
B = foreach A generate (name, age);
DUMP B;
```

Output (results):

```
(joe smith,20)
(amy chen,22)
(leo allen,18)
```

2 Bag Construction

A = load ‘path to hdfs/students.tsv' as (name:chararray, age:int, gpa:float);

B = foreach A generate {(name, age)}, {name, age};

DUMP B;

Output : (generates two bags with same fields)

{(joe smith,20)} {(joe smith),(20)}

{(amy chen,22)} {(amy chen),(22)}

{(leo allen,18)} {(leo allen),(18)}

3 Map Construction

A = load ‘path to hdfs/students.tsv' as (name:chararray, age:int, gpa:float);

B = foreach A generate [name, gpa];

DUMP B;

Output:

[joe smith#3.5]

[amy chen#3.2]

[leo allen#2.1]

Pig Functions

- **Load/Store Functions**
  - Load Data Sets
  - Store Data Sets to HDFS
- **Filter Function**
  - Filter records based on condition
- **Eval Functions**
  - Evaluate values like, Sum, Average, Max, Count etc
- **Group By**
  - Group by
  - Flattern
- **UDFs – User Defined Functions**
  - Custom functions can be defined of either of three types – Load/Store, Eval or Filter

PIG Latin data types

Simple Data Types	Description	Example
Scalars		
int	Signed 32-bit integer	10
long	Signed 64-bit integer	Data: 10L or 10l Display: 10L
float	32-bit floating point	Data: 10.5F or 10.5f or 10.5e2f or 10.5E2F Display: 10.5F or 1050.0F
double	64-bit floating point	Data: 10.5 or 10.5e2 or 10.5E2 Display: 10.5 or 1050.0
Arrays		
chararray	Character array (string) in Unicode UTF-8 format	hello world
bytearray	Byte array (blob)	
Complex Data Types		
tuple	An ordered set of fields.	(19,2)
bag	An collection of tuples.	{(19,2), (18,1)}
map	A set of key value pairs.	[open#apache]

Lets Learn by Solving a Case Study

Case Study

Problem Statement :

ABC Entertainments is a production house and required to finalize roadmap for coming year.

As an entertainment business with major focus on movies production, business needs to make multiple decisions.

For example,

1. For social media marketing what age group, gender or profession people should be targeted ?
2. For a given set of viewers what type of movie is most appreciated, to be able to decide before investing.

Solution:

As we are given with historic data set for movies, its ratings and user's details we can analyze some information from it. For the scope of case study we will consider data set from movieLense labs for all the movies released in April 1998 – approx. 1700 movies with 100 Thousand ratings.

Case Study [Contd.]

Should we choose Hive or Pig for this case ?

- Data sets are provided as flat files.
- It requires to perform some joins, aggregations and filter on data sets to achieve end results.
- Since we are not fully aware of dataset and data quality, we may be interested to explore data initially to learn about data set !
- We might be interested to store intermediate results of our analysis

This problem cane be solved with Hive, Pig or Map Reduce programs also.

As we are interested to store multiple intermediate results – Pig fits well here. With Hive we have to create those many intermediate tables and take care of truncating during each run.

Since there are join, filter and aggregation operation similar can be achieved with Hive. When you will compare Hive solution script with Pig, Pig script is going to look short and simple. It makes it easy scripting for complex analytical jobs.

Since we don't know actual data quality before starting, initial data exploration can be done with Pig, without worrying about data formats and fields within, Since we don't need to create any sort of tables here !

Case Study

Data Set Used for Case Study : Movie Review Data Set from MovieLens Labs.

Stable benchmark dataset. 100,000 ratings from 1000 users on 1700 movies. Released 4/1998.

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Attached : movie-100k.zip

Case Study [Cont.]

Data Directory : movie-100k

There are multiple "|" separated files in this directory, few are described below which will be used in this session.

File Name	Description	Fields in File	Fields separator
u.data	Ratings data	user id, item id, rating, timestamp	Tab
u.Item	Movie data	movie id, movie title, release date, video release date,IMDb URL, unknown, Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir,Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western	|
u.User	User demographic data	user id, age, gender, occupation, zip code	|

Pig Latin Syntax – Before we start using them

Load/Store Function

- dataSetX = LOAD '{HDFS Path}' USING {Load Function Name} AS [{schema Specification - optional}]

Filter Function

- filteredData = FILTER {dataset} BY {Condition}

Group

- groupedDataSet = GROUP {dataset} BY {some field of dataset}
- This returns a Bag, which has first field as Group Key and second fields will be a Bag with all tuples grouped by specified Key.

Loop – Foreach & Generate

- Foreach is an iterator and generate allows to generate new Tuple/Bag/Map with specified fields
- resultDataSet = FOREACH {bag dataset} GENERATE {field1, filed2 ….}
- [see slide 14]

Join

- joinResult = JOIN {dataSet1} BY {commonFieldFromDataSet1}, {dataSet2} BY {commonFieldFromDataSet2}

Case Study [Cont.]

1 Load data in HDFS

- First load movie-100k folder to HDFS from your local file system

We will run Pig in default i.e Hadoop mode and Pig will look for Data on HDFS in that case.

hadoop fs –put /local path to data set/movie-100k /desired-hdfs-path/

- Start Pig in default mode with below command
  - Pig

  This will start pig console called grunt

```
[training@localhost ~]$ pig
2016-12-18 11:05:51,514 [main] INFO  org.apache.pig.Main - Apache Pig version 0.10.0-cdh4.1.1 (r: unknown) compiled Oct 16 2012, 12:20:22
2016-12-18 11:05:51,514 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/training/pig_1482077151512.log
2016-12-18 11:05:51,773 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:8020
2016-12-18 11:05:51,775 [main] WARN  org.apache.hadoop.conf.Configuration - mapred.used.genericoptionsparser is deprecated. Instead, use mapreduce.client.genericoptionsparser.used
2016-12-18 11:05:51,782 [main] WARN  org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-12-18 11:05:52,326 [main] WARN  org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

2 Find Simple Statistic from given Data Set

- **Statistic about User's age**

User data is available in **u.user** file. First thing required is to load this data set

```
users = LOAD '/hdfs-path-to/movie-100k/u.user'  USING  PigStorage('|')   AS
(userId,age,gender,occupation,zipCode);
```

Verify if above step is valid by simply dumping data from data set loaded above

Try: dump users;

Expected output is data from file should be seen on console

Note: All Keywords like LOAD, FILTER are marked in capital letters just to highlight. They are case-insensitive.

Case Study [Cont.]

- **Statistic about User's age [Cont.]**

```
allUsers = GROUP users ALL;
stats = FOREACH allUsers GENERATE COUNT(users), AVG(users.age), SUM(users.age);
dump stats;
```

Expect Total Count of Users, average age of all users and Total Sum of all user's age !

- **Gender distribution**

```
byGender = GROUP users BY gender;
genderStats = FOREACH byGender GENERATE group, COUNT(users), AVG(users.age);
dump genderStats;
```

Expect gender wise count and average age of user in each gender !

Case Study [Cont.]

- **Filter user by profession and find age group wise count of users in particular profession**

  Lets filter programmers !

```
programmers = FILTER users BY occupation == 'programmer';
progsByAge = GROUP programmers BY age;
progCountsByAge = FOREACH progsByAge GENERATE group AS age, COUNT(programmers) as NumProgs;
progsCountsByAgeSorted = ORDER progCountsByAge BY NumProgs DESC;
```

Try

```
dump progsCountsByAgeSorted;
```

Dump is good to show results on screen. How do I persist result on Disk ?

Use Store Function to Store dataset into HDFS file!

Case Study [Cont.]

- **Store Data sets in HDFS Files**

```
STORE stats INTO ‘hdfs-path-to-non-existing-dir’ USINg PigStorage('\t');

STORE genderStats INTO ‘hdfs-path-to-non-existing-dir’ USING
PigStorage('\t');

STORE progsCountsByAgeSorted INTO ‘hdfs-path-to-non-existing-dir’ USING
PigStorage('\t');
```

Case Study [Cont.]

3 Find Top 25 Rated Movies

- Load ***Votes*** data set

```
votes = LOAD '/user/training/movie-100k/u.data' USING PigStorage('\t') AS
(userId,itemId,rating,timestamp);
```

- Load ***movies*** data set

```
movies = LOAD '/user/training/movie-100k/u.item' USING PigStorage('|') AS
(movieId,movieTitle, releaseDate, videoReleaseDate, imdbURL, unknown, Action,
Adventure, Animation, Childrens, Comedy, Crime, Documentary, Drama, Fantasy, FilmNoir,
Horror, Musical, Mystery, Romance, SciFi,Thriller,War,Western);
```

Case Study [Cont.]

- **Group votes by Item id /movie id**

  movieVotesGroup = GROUP votes BY itemId;

- **Flattern above generated relation and generate average rating, count of votes for a movie id**

  movieVotes = FOREACH movieVotesGroup GENERATE FLATTEN(group) AS movieId,
  AVG(votes.rating) AS avgRating, COUNT(votes) AS numVotes;

- **Join movieVotesGroup and movieVotes data sets**

  moviesWithVotes = JOIN movieVotes BY movieId, movies BY movieId;

- **Sort**

  moviesWithVotesSorted = ORDER moviesWithVotes BY movieVotes::avgRating DESC;

- **Get Top 25**

  top25 = LIMIT moviesWithVotesSorted 25;

- **Store Result**

  STORE top25 INTO 'some-hdfs-dir' USING PigStorage('\t');

Case Study [Cont.]

- **Some Practice Assignments on same data set**
  - How many are between 35 and 59
  - How many of these are females
  - Find eldest user who is not programmer or doctor

User Defined Functions (UDFs)

Pig provides extensive support for user defined functions (UDFs) as a way to specify custom processing.

Pig UDFs can currently be implemented in three languages: Java, Python, and JavaScript.

Pig also provides support for Piggy Bank, a repository for JAVA UDFs.

Type of UDFs

- Eval Functions
- Filter Functions
- Load Functions
- Store Functions

User Defined Functions (UDFs) [Contd.]

1) All user defined Evaluation functions extend “org.apache.pig.EvalFunc”

2) All functions must override “exec” method.

3) “pig-0.10.0-cdh4.1.1-withouthadoop” make sure this file is in your build path apart from all other files that were mentioned in previous class .

This file is present in “usr/lib/pig/”

Other files are in “usr/lib/hadoop/client-0.20”

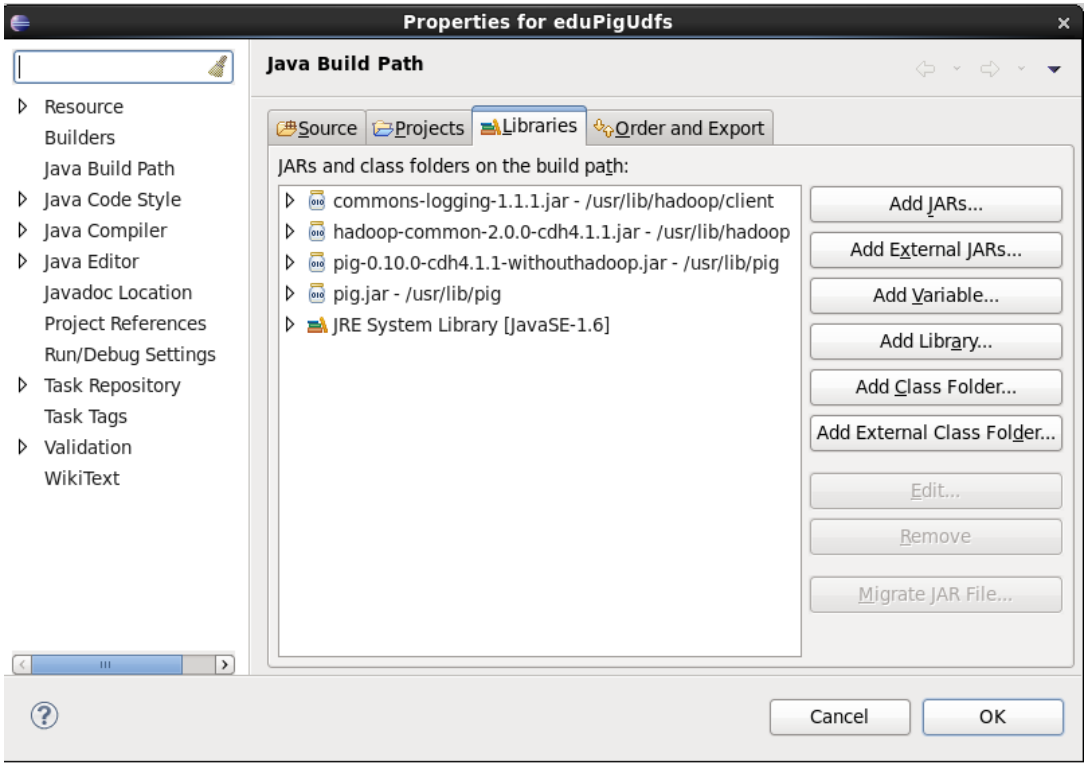
Filter Function For our User Data Set

Lets says, we would like to filter user who are in their 30s and in any technical profession.

Although, It's a simple example, Idea is to understand Custom Filter Function.

We will write Custom Filter Function 'FilterTechProfessionalsInAgeOf30s' in Java.

Libraries added for UDF



Java Eclipse Project for UDF

Source Code and Pig Script using UDF


eduPigUdfs.zip


pig_UserUdf.pig

Export Project as JAR file and Refer below Pig Script to Use UDF

Name of Class becomes Name of Function prefixed with Package name if any.

In Grunt >

users = LOAD '*your u.user file path on HDFS*' USING PigStorage('|') AS (userId, age, gender, occupation: chararray, zipCode);

REGISTER /Your Local Path to/ExportedUdf.jar;

filtered = FILTER users BY **com.edupristine.training.pig.FilterTechProfessionalsInAgeOf30s();**

dump filtered;

Java UDF Code for Custom Filter Condition implementation

```
package com.edupristine.training.pig;

import java.io.IOException;

import org.apache.pig.FilterFunc;
import org.apache.pig.data.Tuple;

public class FilterTechProfessionalsInAgeOf30s extends FilterFunc {

    @Override
    public Boolean exec(Tuple input) throws IOException {
        String profession = input.get(3).toString();
        int age = Integer.parseInt(input.get(1).toString());
        boolean shouldFilter = false;
        if ((profession.contains("technician") || profession.contains("programmer") ||
                profession.contains("engineer")) && (age >= 30 && age < 40)) {
            shouldFilter = true;
        }
        return shouldFilter;
    }

}
```

Running Pig Script

In real scenario, You are likely to develop your solution by trying out each steps and write next statement. Once you have solution script tested and ready, You would like to schedule it or run it as a batch job. Just put all your pig statements in one ".pig" file and run as a script.

For example : in your command prompt

```
$>  pig pig_UserUdf.pig
```

Pig Developer's tool

- **IILLUSTRATE -** Use the ILLUSTRATE operator to review how data is transformed through a sequence of Pig Latin statements. know more

- **DESCRIBE -** to review the schema of a particular alias. know more

- **EXPLAIN -** Displays execution plans. know more

PIG Resources …

Apache

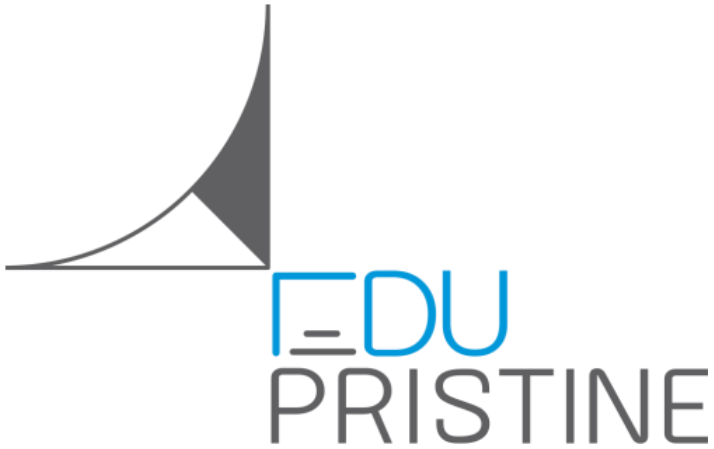
https://pig.apache.org/

Yahoo Developer Network

http://developer.yahoo.com/hadoop/tutorial/pigtutorial.html

Horton Works

http://hortonworks.com/hadoop/pig/



Thank You!

help@edupristine.com

www.edupristine.com