Pig
A language for data processing in Hadoop

Antonino Virgillito

THE CONTRACTOR IS ACTING UNDER A FRAMEWORK CONTRACT CONCLUDED WITH THE COMMISSION

Apache Pig: Introduction

- Tool for querying data on Hadoop clusters
- Widely used in the Hadoop world
  - Yahoo! estimates that 50% of their Hadoop workload on their 100,000 CPUs clusters is genarated by Pig scripts
- Allows to write data manipulation scripts written in a high-level language called Pig Latin
  - Interpreted language: scripts are translated into MapReduce jobs
- Mainly targeted at joins and aggregations

Overview of Pig

- PigLatin
  - **Language for definition of data flow**
- Grunt
  - **Interactive shell for typing and executing PigLatin statements**
- Interpreter and execution engine

Execution of Pig Script

- Grunt
  - **Interactive execution**
  - **Each statement is interpreted as it is typed**
  - **Execution is delayed until output is requested**
  - **Useful for debugging and ad hoc data inspection**
- Pig Editor in Hue
  - **Assistance to script development: autocompletion, language reference, syntax highlighting, navigator**
  - **Allows only batch execution of the whole script**

Concepts

- A script in Pig allows to define flows of data manipulation over datasets stored in HDFS
  - **Sequence of statements**
- Each dataset is organized in:
  - **Fields -> equivalent to columns**
  - **Tuples -> equivalent to rows (collection of fields)**
  - **Bags -> equivalent to tables (collection of tuples)**
- Typically, a Pig Latin script starts by loading one or more datasets into bags, and then creates new bags by modifying those it already has

Pig Commands

Loading datasets from HDFS

HDFS Path to the file

Separator

New bag for the data

```
users = load 'Users.csv' using PigStorage(',') as
(username: chararray, age: int);

pages = load 'Pages.csv' using PigStorage(',') as
(username: chararray, url: chararray);
```

Can specify the schema of the bag

Pig Commands

Loading datasets from HDFS

```
users = load 'Users.csv' using PigStorage('') as
(rowastext: chararray);
```

A same file can be considered as a bag with a different schema, simply by changing the separator

This allows to use Pig also for data preparation and pre-processing

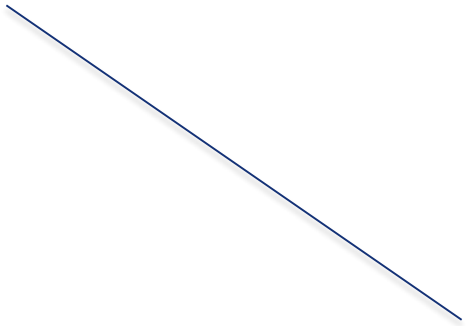
Interactive execution

- Special commands are available in Grunt to preview the results of each command
- **DESCRIBE**: reviews the schema of a bag
- **ILLUSTRATE**: displays the result of a step-by-step executions of statements using a tiny subset of data
- **EXPLAIN**: displays the execution plan

Pig Commands

Filtering data

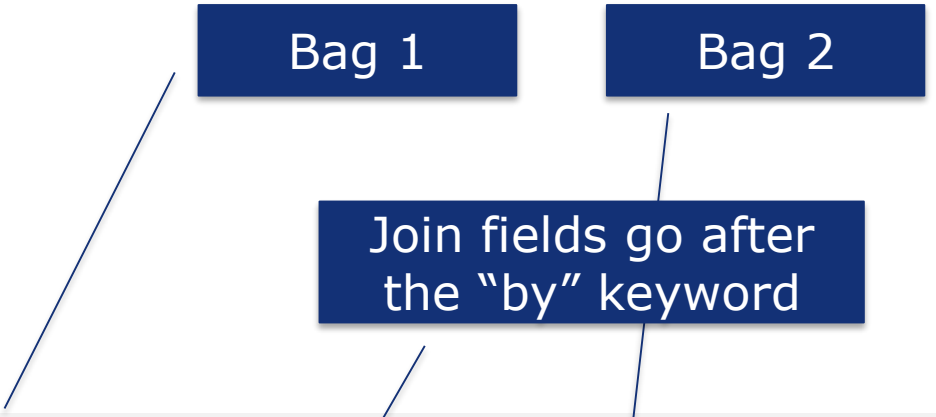
```
users_1825 = filter users by age>=18 and age<=25;
```



Creates a new bag with data from the first bag filtered by age

Pig Commands

Join datasets




```
joined = join users_1825 by username, pages by username;
```

Creates a new bag with tuples from the two joined bags

```
DESCRIBE joined;
joined: {user_1825::username: chararray, user_1825::age: int, pages::username: chararray, pages::url: chararray}
```

Adds the names of the original bag to the field names

Aggregation functions

- Two statements are required to apply an aggregation function to a field in a bag, like a count or sum
- First, a call to GROUP creates a bag with nested tuples, where all the tuples belonging to a same group are collected within a same tuple
- Then, a call to FOREACH ensures that the aggregation function is applied to all the elements of the group

Pig Commands

Group records

```
grouped = group joined by url;
```

Creates a new dataset with an element named *group* and another named *joined*. The former corresponds to the grouping field, the latter is a bag with all the fields in the grouped dataset:

```
DESCRIBE grouped;

grouped: {group: chararray, joined:(username:chararray,age:int)}
```

«url» field is renamed as «group»

Pig Commands

Group records

```
grouped = group joined by url;
```

The new dataset will have one record for each distinct url, with all the records corresponding to that url grouped in the bag:

```
dump grouped;

(www.twitter.com, {(alice, 15), (bob, 18)})
(www.facebook.com, {(carol, 24), (alice, 14), (bob, 18)})
```

Pig Commands

Applying a function to records in a dataset

```
summed = foreach grouped generate group as url,
COUNT(joined) AS views;
```

The foreach statement is used to apply an aggregation function to all elements of a grouped dataset

```
dump summed;

(www.twitter.com, 2)
(www.facebook.com, 3)
```

Pig Commands

Counting all the records in a dataset

```
joined_all = group joined all;
joined_cnt = foreach joined_all generate COUNT(joined);
```

The GROUP ALL statement produces a dataset with a single row where all the rows of the original dataset are grouped in the same bag.

```
dump joined_cnt;

5
```

Pig Commands

Sort a dataset

```
sorted = order summed by views desc;
```

Filter first n rows

```
top_5 = limit sorted 5;
```

Built-in Functions and operators

- General operators
  - **Arithmetic, comparison, dereference, FLATTEN…**
- Relational operators
  - **GROUP, JOIN, COGROUP, CROSS, SPLIT…**
- Eval functions (aggregations)
  - **AVG, SUM, COUNT, CONCAT…**
- Math functions
- String functions
- Bag and Tuple functions
  - **TOBAG, TOP, TOTUPLE**

https://pig.apache.org/docs/r0.15.0/

User-Defined Functions

- Custom processing can be achieved by extending Pig with User-defined functions (UDFs)
- UDFs can be developed in different programming languages
  - Java, Jython, Python, JavaScript, Ruby and Groovy
- Functions written in Java have the most extensive support, allowing to customize all parts of the processing

http://pig.apache.org/docs/r0.15.0/udf.html

Generating the output

- Statements that generate the output once executed in interactive mode trigger the execution of the MapReduce jobs associated to the data flow that produced the bag
- DUMP x
  - **Writes the content of the bag x to console**
- STORE x INTO file
  - **Writes the content of the bag x in a file in HDFS**

Pig Commands

Writes a dataset to HDFS

```
store top_5 into 'top5_sites.csv';
```

Choosing the Right Tool

- Choose the best solution for the given task – Mix and match as needed
- MapReduce
  - **Low/level approach offers flexibility, control, and performance**
  - **More time-consuming and error-prone to write**
  - **Choose when control and performance are most important**
- Pig and Hive
  - **Faster to write, test, and deploy than MapReduce**
  - **Better choice for most analysis and processing tasks**

Choosing the Right Tool

- Use Hive or Pig when…
  - **You need support for custom file types, or complex data types**
- Use Pig when…
  - **You have developers experienced with writing scripts**
  - **You need complex processing flows**
  - **Your data is unstructured-semi/structured**
- Use Hive When…
  - **You have very complex long/running queries**

Pig/Hive vs. RDBMS

- Not intended to replace a RDBMS
- Relational databases are optimized for:
  - **small to medium amounts of data**
  - **Immediate results**
  - **In/place modification of data**
- Pig and Hive are optimized for:
  - **Large amounts of read-only data**
  - **Extensive scalability at low cost**
- Pig and Hive are better suited for batch processing
- RDBMSs are better for interactive use