LEARNING apache-pig

#apache-pig

Table of Contents

About

Chapter 1: Getting started with apache-pig

Remarks

This section provides an overview of what apache-pig is, and why a developer might want to use it.

It should also mention any large subjects within apache-pig, and link out to the related topics. Since the Documentation for apache-pig is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Linux

Requirements (r0.16.0)

Mandatory

As per current `Apache-Pig` documentation it supports only `Unix` & `Windows` operating systems.

- Hadoop 0.23.X, 1.X or 2.X
- Java 1.6 or Later versions installed and JAVA_HOME environment variable set to Java installation directory

Optional

- Python 2.7 or more (Python UDFs)
- Ant 1.8 (for builds)

Download the latest Pig release

Download the latest version of pig from http://pig.apache.org/releases.html#Download

Installation

```
mkdir Pig
cd Downloads/
tar zxvf pig-(latest-version).tar.gz
tar zxvf pig-(latest-version).tar.gz
mv pig-(latest-version).tar.gz/* /home/Pig/
```

Configuration

After installing Apache Pig, we have to configure it.

Open the .bashrc file

```
vim ~/.bashrc
```

In the .bashrc file, set the following variables –

```
export PIG_HOME = /home/Pig
export PATH  = PATH:/home/Pig/bin
```

save the file and reload bashrc again in the environment using

```
. ~/.bashrc
```

Verifying Pig version

```
pig -version
```

If the installation is successful, the above command displays the installed Pig version number.

Testing Pig Installation

```
pig -h
```

This should display all the possible commands associated with pig

Your pig is now installed locally and you can run it using local parameter like

```
pig -x local
```

Connecting to Hadoop

If Hadoop1.x or 2.x is Installed on the cluster and the HADOOP_HOME environment variable is setup.

you can connect pig to Hadoop by adding the line in the .bashrc like before

```
export PIG_CLASSPATH = $HADOOP_HOME/conf
```

Running Pig

Execution Modes

You can run Pig either using the `pig` *(bin/pig)* command or by running `jar` file *(java -cp pig.jar)*

`PIG` scripts can be executed in 3 different modes:

- **Local Mode**

```
pig -x local ...
```

- **Mapreduce Mode** (default mode)

```
pig -x mapreduce ...
     (or)
pig ...
```

- **Tez Local Mode**

```
pig -x tez ...
```

Interactive Mode

Pig can be run in interactive mode using the `Grunt` shell. Pig Latin statements and commands can be entered interactively in this shell.

Example

```
$ pig -x <mode> <enter>
grunt>
```

`Mode` can be one of execution modes as explained in the previous section.

Batch Mode

Pig can also be executed in batch mode. Here a `.pig` file containing a list of pig statements and commands is provided.

Example

```
$ pig -x <mode> <script.pig>
grunt>
```

Similarly `Mode` can be one of execution modes as explained in the previous section.

Word Count Example in Pig

Input file

```
Mary had a little lamb
its fleece was white as snow
and everywhere that Mary went
the lamb was sure to go.
```

Pig Word Count Code

```
-- Load input from the file named Mary, and call the single
-- field in the record 'line'.
```

```
input = load 'mary' as (line);

-- TOKENIZE splits the line into a field for each word.
-- flatten will take the collection of records returned by
-- TOKENIZE and produce a separate record for each one, calling the single
-- field in the record word.
words = foreach input generate flatten(TOKENIZE(line)) as word;

-- Now group them together by each word.
grpd = group words by word;

-- Count them.
cntd = foreach grpd generate group, COUNT(words);

-- Print out the results.
dump cntd;
```

Output

```
Mary,2
had,1
a,1
little,1
lamb,2
its,1
fleece,1
was,2
white,1
as,1
snow,1
and,1
everywhere,1
that,1
went,1
the,1
sure,1
to,1
go,1
```

What Is Pig?

Pig provides an engine for executing data flows in parallel on Hadoop. It includes a language, Pig Latin, for expressing these data flows. Pig Latin includes operators for many of the traditional data operations (join, sort, filter, etc.), as well as the ability for users to develop their own functions for reading, processing, and writing data. Pig is an Apache open source project. This means users are free to download it as source or binary, use it for themselves, contribute to it, and—under the terms of the Apache License—use it in their products and change it as they see fit.

Read Getting started with apache-pig online: https://riptutorial.com/apache-pig/topic/3244/getting-started-with-apache-pig

Chapter 2: Cube Operation

Examples

Basic CUBE

Considering the following case:

We have user event data with 4 dimensions:

1. A/B Test bucket (prod/test)
2. client Type (web/mobile)
3. module (order/report)
4. event (click/view)

```
test    mobile    order_module    click
prod    web    order_module    view
prod    mobile    order_module    click
```

A report system might want to report metrics for different combination, such as:

1. what's the total **click** count from **mobile** user?
2. what's the total **click** count from different test bucket?
3. what's the total **web** page **view** for **order_module** in **prod** bucket?

As you can see there are many combinations. It's not quite time efficient if we only store smallest granularity metrics and then roll them up when receiving a query. So one solution is to **pre-compute ALL combinations**.

Here's how we could do that with PIG's CUBE operation.

```
example = LOAD './cube.example' AS (product:chararray, client:chararray, module:chararray,
action:chararray);

cubed_data = CUBE example BY CUBE(product, client, module, action);

final_data = FOREACH cubed_data GENERATE $0, COUNT_STAR($1);

dump final_data;
```

It will produce output of all combinations and total counts. See the output of previous dump -- with this stats, we could answer previous questions with direct answer. No further aggregation needed.

```
((prod,web,order_module,view),1)
((prod,web,order_module,),1)
((prod,web,,view),1)
((prod,web,,),1)
((prod,mobile,order_module,click),1)
((prod,mobile,order_module,),1)
```

```
((prod,mobile,,click),1)
((prod,mobile,,),1)
((prod,,order_module,view),1)
((prod,,order_module,click),1)
((prod,,order_module,),2)
((prod,,,view),1)
((prod,,,click),1)
((prod,,,),2)
((test,mobile,order_module,click),1)
((test,mobile,order_module,),1)
((test,mobile,,click),1)
((test,mobile,,),1)
((test,,order_module,click),1)
((test,,order_module,),1)
((test,,,click),1)
((test,,,),1)
((,web,order_module,view),1)
((,web,order_module,),1)
((,web,,view),1)
((,web,,),1)
((,mobile,order_module,click),2)
((,mobile,order_module,),2)
((,mobile,,click),2)
((,mobile,,),2)
((,,order_module,view),1)
((,,order_module,click),2)
((,,order_module,),3)
((,,,view),1)
((,,,click),2)
((,,,),3)
```

Read Cube Operation online: https://riptutorial.com/apache-pig/topic/6120/cube-operation

Chapter 3: LOAD Operator

Examples

Loading Stock market data

Let us assume the following stock market data stored in `HDFS`. It is a `csv` file with fields: **Symbol, Date, Open, High, Close & Volume**.

```
ABT,20160106,42.310001,42.98,42.209999,42.560001,5906000
BAC,20160201,14.05,14.09,13.8,13.96,105739400
CAS,20160129,1.9,1.97,1.83,1.84,34500
DCA,20160129,3.46,3.54,3.46,3.51,84600
ECL,20160114,103.480003,105.400002,102.480003,104.82,1485000
FAF,20160201,34.040001,34.82,33.939999,34.639999,1222600
TYL,20160201,156.070007,159.550003,155.690002,158.259995,177100
UTL,20160201,38.610001,39.889999,38.57,39.27,119500
VTR,20160128,54.09,54.73,53.549999,53.790001,2441300
WWE,20160201,17.629999,18,17.27,17.799999,734100
XRX,20160104,10.41,10.43,10.13,10.3,9122600
YUM,20160104,71.32,72.25,70.639999,72.209999,3466300
ZTR,20160104,12.1,12.14,11.98,12.11,60200
```

Example 1 A simple `LOAD` statement for the above data would look like:

```
stocks = load '/user/pig/stock.txt' using PigStorage(',') as
             (sym:chararray, date:int, open:float, high:float, low:float,
              close:float, vol:int);
```

Loading data from ElasticSearch

Before checking the specific syntax, let's take a look on how to setup your environment to load needed plugins.

Setup

To load data directly from ElasticSearch you need to download the `elasticsearch-hadoop` plugin. You have different ways to make it work, for a quick setup you can do the following.

In order to make it work, you need to put the jar file `elasticsearch-hadoop-<version>.jar` in a folder of the node where you have the pig server installed. In my case - quite common googling around - I also needed to add `commons-httpclient-<version>.jar` inside that folder.

Then you can run `pig` in shell mode (called `grunt`) simply typing `pig` on the console. Then you have to load those two jars in the following way:

```
REGISTER /path/to/jars/commons-httpclient-<version>.jar;
REGISTER /path/to/jars/elasticsearch-hadoop-<version>.jar;
```

Now you are ready to write some code.

Example

Let's check the syntax to load data from a complex case.

```
DATA = LOAD 'my_index/log' USING org.elasticsearch.hadoop.pig.EsStorage(
'es.nodes=https://server1:port1,https://server2:port2,https://server3:port3',
'es.query=?q=*',
'es.net.ssl=true',
'es.net.http.auth.user=user',
'es.net.http.auth.pass=pass',
'es.net.ssl.keystore.type=JKS',
'es.net.ssl.truststore.location=file:///path/to/truststore.jks',
'es.net.ssl.truststore.pass=pass');
```

This is the complete example, now let's analyze it step by step.

- `es.nodes` holds the list of the nodes of your ElasticSearch cluster. You have to specify your nodes as a comma separated list, with the associated port.
- `es.query` holds the query that will be submitted to ElasticSearch in order to fetch data. You can also put a query in DSL format, but be careful that only the match part of the query will be considered! If you try to limit the number of fields through the query DSL it won't work: in order to achieve that you need to use the `es.read.source.filter` parameter. example of a query DSL: `'es.query = { "query":{ "match_all":{} } }'`
- `es.net.ssl=true` is self explanatory, you need also to give the login credentials to ElasticSearch with `es.net.http.auth.user` and `es.net.http.auth.pass`.
- `es.net.ssl.keystore.type` if you need a truststore, you can select here the type. In the `es.net.ssl.truststore.location` parameter you set the location of the file, be careful to add `file://` prefix, and in the `es.net.ssl.truststore.pass` parameter you set the password of the truststore file.

Some useful settings

- `es.read.source.filter=field1,field2,field3` allows you to fetch only the specified fields from ElasticSearch (in this example three).
- `es.output.json=true` allows you to fetch data in a key-value format (JSON). Setting to `false` will return data in CSV format (default).

Read LOAD Operator online: https://riptutorial.com/apache-pig/topic/7257/load-operator

Credits

S. No	Chapters	Contributors
1	Getting started with apache-pig	Bhavesh, Brian Armstrong, Community, DhiwaTdG, Mzzzzzz, pratiklodha
2	Cube Operation	Wenzhong
3	LOAD Operator	Andrea Romagnoli, DhiwaTdG