pig_params_azure

July 1, 2023

1 PIG et Paramètres sur Azure - énoncé

Manipulation de données JSON en Map/Reduce avec PIG sur HDInsight.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Paramètres

Les sites web produisent des données en continu. On utilise fréquemment le même script pour traiter les donnéesd'un jour, du lendemain, de jour d'après… Tous les jours, on veut récupérer la fréquentation de la veille. La seule chose qui change est la date des données qu'on veut traiter. Plutôt que de recopier un script en entier pour changer une date qui apparaît parfois à plusieurs endroits, il est préférable d'écrire un script ou la date apparaît comme une variable.
Ce notebook va illustrer ce procédé sur la construction d'un histogramme. Le paramètre du script sera la largeur des barres de l'histogramme (ou bin en anglais).

1.2 Connexion au cluster

On prend le cluster Cloudera. Il faut exécuter ce script pour pouvoir notifier au notebook que la variable `params` existe.

[2]:
```
import pyensae
from pyquickhelper.ipythonhelper import open_html_form
params={"blob_storage":"", "password1":"", "hadoop_server":"", "password2":"",␣
↪"username":"alias"}
open_html_form(params=params,title="server + hadoop + credentials", key_save="blobhp")
```

[2]:
```
<IPython.core.display.HTML at 0x7c801d0>
```

[3]:
```
import pyensae
%load_ext pyensae
%load_ext pyenbc
blobstorage = blobhp["blob_storage"]
blobpassword = blobhp["password1"]
hadoop_server = blobhp["hadoop_server"]
hadoop_password = blobhp["password2"]
username = blobhp["username"]
client, bs =  %hd_open
client, bs
```

[3]:
```
(<pyensae.remote.azure_connection.AzureClient at 0xabaea50>,
 <azure.storage.blobservice.BlobService at 0xabaea90>)
```

1.3 Upload version

On commence par simuler des données.

[4]:
```
import random
with open("random.sample.txt", "w") as f :
    for i in range(0,10000) :
        x = random.random()
        f.write(str(x)+"\n")
```

On uploade le fichier sur le cluster (il n'est pas besoin de créer le répertoire au préalable).

[5]:
```
%blob_up random.sample.txt /$PSEUDO/random/random.sample.txt
```

[5]:
```
'$PSEUDO/random/random.sample.txt'
```

[6]:
```
%blob_ls /$PSEUDO/random
```

[6]:
```
                                   name                  last_modified  \
0  xavierdupre/random/random.sample.txt  Thu, 27 Nov 2014 23:21:26 GMT

                content_type  content_length  blob_type
0  application/octet-stream          202619  BlockBlob
```

1.4 PIG et paramètres

On indique un paramètre par le symbole : `$bins`. La valeur du paramètre est passé sous forme de chaîne de caractères au script et remplacée telle quelle dans le script. Il en va de même des constantes déclarées grâce au mot-clé %declare.
La sortie du script inclut le paramètre : cela permet de retrouver comment ces données ont été générées.

[7]:
```
%%PIG histogram.pig

values = LOAD '$CONTAINER/$PSEUDO/random/random.sample.txt' USING PigStorage('\t') AS␣
 ↪(x:double);

values_h = FOREACH values GENERATE x, ((int)(x / $bins)) * $bins AS h ;

hist_group = GROUP values_h BY h ;

hist = FOREACH hist_group GENERATE group, COUNT(values_h) AS nb ;

STORE hist INTO '$CONTAINER/$PSEUDO/random/histo_$bins.txt' USING PigStorage('\t') ;
```

Pour supprimer les précédents résultats :

[8]:
```
if client.exists(bs, client.account_name, "$PSEUDO/random/histo_0.1.txt"):
    r = client.delete_folder (bs, client.account_name, "$PSEUDO/random/histo_0.1.txt")
    print(r)
```

On exécute le job. Comme la commande magique supportant les paramètres n'existe pas encore, il faut utiliser la variable `client` et sa méthode pig_submit qui fait la même chose. Elle upload le script puis le soumet.

[9]:
```
jid = client.pig_submit(bs, client.account_name, "histogram.pig", params =␣
↪dict(bins="0.1"), stop_on_failure=True )
```

[9]:
```
{'id': 'job_1416874839254_0101'}
```

[10]:
```
st = %hd_job_status jid["id"]
st["id"],st["percentComplete"],st["status"]["jobComplete"]
```

[10]:
```
('job_1416874839254_0101', '100% complete', True)
```

[11]:
```
%hd_tail_stderr jid["id"]
```

[11]:
```
<IPython.core.display.HTML at 0x7f31db0>
```

On vérifie que tout s'est bien passé. La taille devrait être équivalent à l'entrée.

[12]:
```
%blob_ls /$PSEUDO/random
```

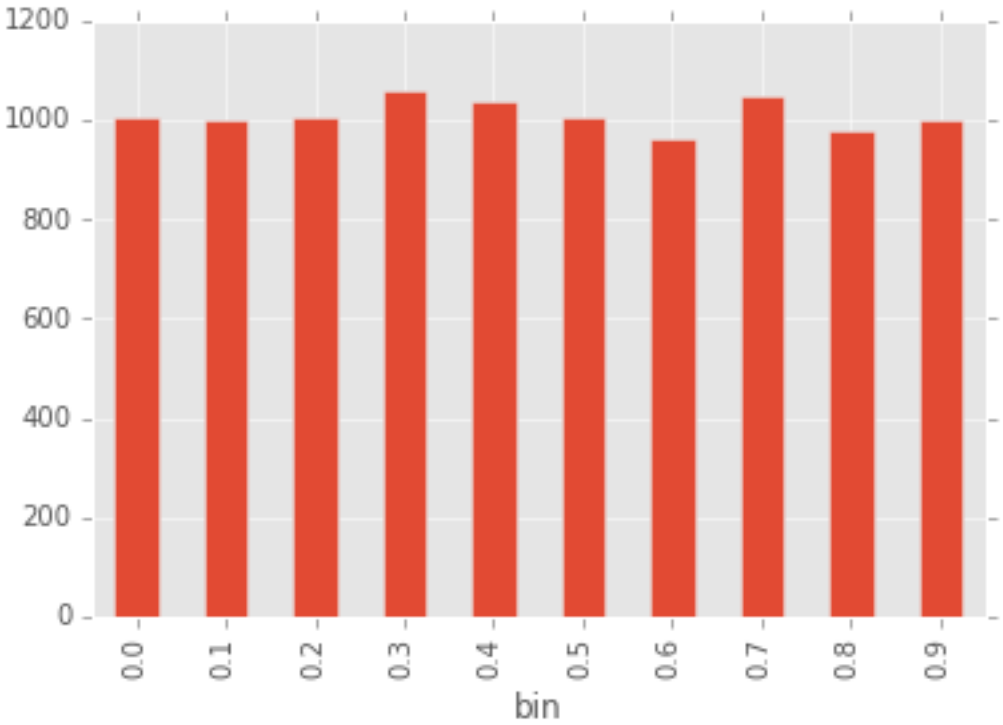
[12]:
```
                                            name  \
0               xavierdupre/random/histo_0.1.txt
1      xavierdupre/random/histo_0.1.txt/_SUCCESS
2  xavierdupre/random/histo_0.1.txt/part-r-00000
3           xavierdupre/random/random.sample.txt

                   last_modified              content_type  content_length  \
0  Thu, 27 Nov 2014 23:28:55 GMT                                         0
1  Thu, 27 Nov 2014 23:28:55 GMT  application/octet-stream               0
2  Thu, 27 Nov 2014 23:28:54 GMT  application/octet-stream             131
3  Thu, 27 Nov 2014 23:21:26 GMT  application/octet-stream          202619

   blob_type
0  BlockBlob
1  BlockBlob
2  BlockBlob
3  BlockBlob
```

[13]:
```
import os
if os.path.exists("histo.txt") : os.remove("histo.txt")
%blob_downmerge /$PSEUDO/random/histo_0.1.txt histo.txt
```

[13]:
```
'histo.txt'
```

[14]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import pandas
df = pandas.read_csv("histo.txt", sep="\t",names=["bin","nb"])
df.plot(x="bin",y="nb",kind="bar")
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x9b21d30>
```



1.5 Exercice 1 : min, max

Ajouter deux paramètres pour construire l'histogramme entre deux valeurs `a`,`b`.

[15]:

```
```