

Déployez un modèle dans le cloud

Projet 8 : Parcours Data Scientist

Matthieu Gschwend





Présentation de la problématique

Mise en contexte

Data Scientist dans une très jeune start-up de l'AgriTech, nommée "Fruits!", qui cherche à proposer des solutions innovantes pour la récolte des fruits.

Votre start-up souhaite dans un premier temps se faire connaître en mettant à disposition du grand public une application mobile qui permettrait aux utilisateurs de prendre en photo un fruit et d'obtenir des informations sur ce fruit.

Objectif de notre mission

- *Nous sommes chargés de nous approprier les travaux réalisés par l'alternant :*
Mettre en place un environnement Big Data dans le but d'effectuer les premières étapes de traitement de notre jeu de données (contenant des images de fruits et les labels associés), une extraction de feature et une réduction de dimension.



Sommaire

1. Écosystème du Big Data
 - a. *Contexte Big Data*
 - b. *Calcul distribué*
2. Création de l'environnement Big Data
 - a. *Stockage des données sur S3*
 - b. *Configuration de EMR*
3. Analyse de code
 - a. *Preprocessing*
 - b. *Transfert learning*
 - c. *Réduction de dimension*
4. Démonstration d'exécution du script sur le Cloud



Les enjeux auquel nous devons pouvoir répondre sont les suivants :

- Le *volume* de données
- La *vélocité* à laquelle nous parviennent les données
- La *variété* des formats






Écosystème du Big Data

Calcul distribué : Introduction

Parallèle : les différents threads d'une même machine sont exécutés et partagent une mémoire commune.

Distribué : Les noeuds distants les uns des autres, envoient des messages pour communiquer

Avantages :

- Le passage à l'échelle s'effectue de manière **horizontale** (ajouter des machines supplémentaires).
Dans le modèle parallèle, on passe à l'échelle de manière verticale, en augmentant la puissance des processeurs.
- Grande tolérance aux pannes.
Noeud hs  envoie sa tâche sur un autre.



Écosystème du Big Data

Calcul distribué : Map Reduce

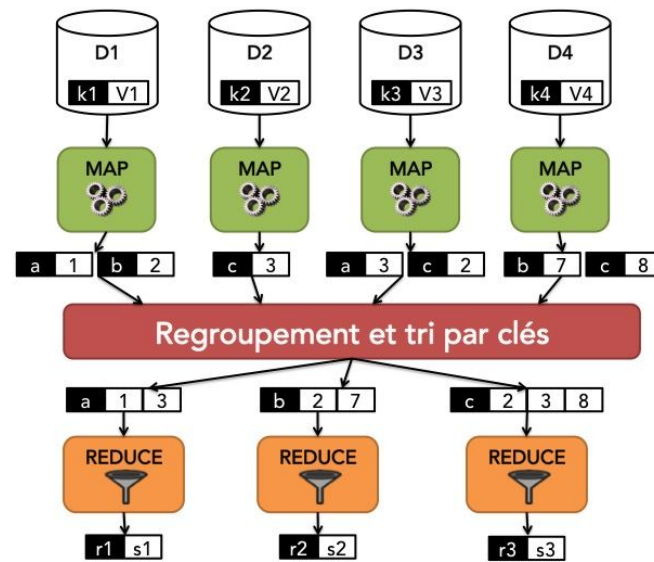
C'est un modèle de programmation, il donne un cadre pour automatiser le calcul en **parallèle** sur des données massives.

Diviser pour mieux régner :

1. **Diviser** : découper le problème initial en sous-problèmes;
2. **Régner** : résoudre les sous-problèmes indépendamment soit de manière récursive, soit directement s'ils sont de petite taille;
3. **Combiner** : construire la solution du problème initial en combinant les solutions des différents sous-problèmes.

Deux opérations majeures :

1. **map** : appliquer une même fonction à tous les éléments de la liste
2. **reduce** : applique une fonction récursivement à une liste et retourne un seul résultat



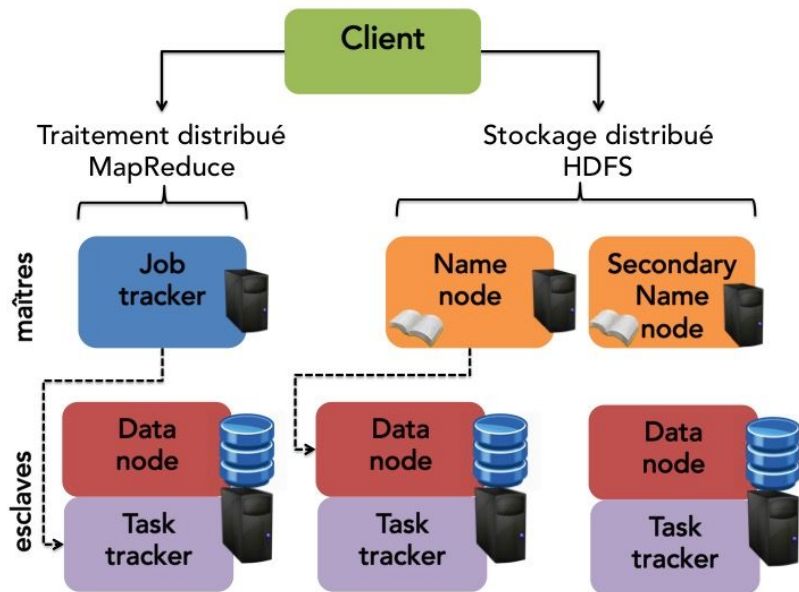


Écosystème du Big Data

Calcul distribué : Hadoop

Le modèle de programmation mapreduce doit pouvoir fonctionner dans un contexte Big Data. Pour que cela soit possible il faut qu'il soit associé à **une infrastructure logicielle** dédiée.

- l'optimisation des transfert disques et réseau en limitant les déplacements de données
- la scalabilité, adapter la puissance au besoin
- tolérance aux pannes



name node contient et stocke tous les noms et blocs des fichiers ainsi que leur localisation dans le cluster
secondary name node sert de namenode de secours

Le **job tracker** est en charge de planifier l'exécution des tâches et de les distribuer sur des **task trackers**



Écosystème du Big Data

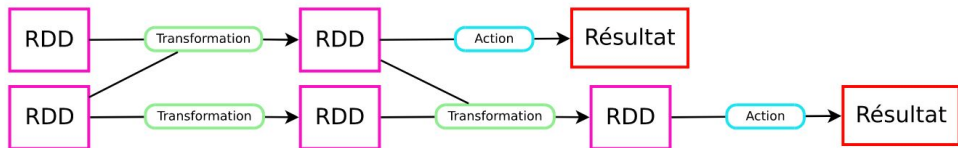
Calcul distribué : Spark

Spark est une alternative à Hadoop MapReduce. Permet de résoudre les problèmes suivants :

- Après une opération map ou reduce, le résultat doit être écrit sur disque
→ opération coûteuse en temps.
- Expressions composées exclusivement d'opérations map et reduce
→ difficile d'exprimer des opérations complexes.

*Spark permet le stockage en mémoire vive (RAM), ce qui donne une accélération des temps de traitement d'un **facteur 10 à 100***

Dans une application Spark, les transformations et les actions réalisées sur les RDD permettent de construire un **graphe acyclique orienté**



Les nœuds sont les RDD et les résultats. Lorsqu'un nœud devient indisponible, il peut être régénéré à partir de ses nœuds parents.



Tolérance aux pannes



Création de l'environnement Big Data

Présentation de AWS

Louer à des tiers des ressources matérielles pour une durée déterminée.

- La capacité de calcul (des serveurs)
- La capacité de stockage (de l'espace disque)

Avantages importants :

- Délégation de la mise en place, de l'entretien et du renouvellement du matériel
- *Élasticité* : possibilité d'agrandir ou de diminuer sa capacité pour des durées variables





Création de l'environnement Big Data

Stockage des données sur S3

S3 est la solution idéale pour stocker nos données :

- Peu onéreuse
- Proche des serveurs de calcul

Transférer les données directement aux serveurs de calculs mais :

- Coût plus élevé
- Les données sont détruite après l'arrêt des serveurs

Etapes :

- Création d'un bucket : commande 'mb'
- Envoi du dossier image : commande 'cp'

```
mat@LAPTOP-2J2CR7I8:~/aws$ aws s3 ls s3://oc-projet8-test
PRE Results/
PRE jupyter/
PRE sample_data_train/
2023-01-31 23:43:02 347 bootstrap-emr.sh
```

<input type="checkbox"/>	apple_6/	Dossier
<input type="checkbox"/>	apple_braeburn_1/	Dossier
<input type="checkbox"/>	apple_crimson_snow_1/	Dossier
<input type="checkbox"/>	apple_golden_1/	Dossier

Contraintes du RGPD

```
mat@LAPTOP-2J2CR7I8:~/aws$ aws s3api get-bucket-location --bucket oc-projet8-test
{
  "LocationConstraint": "eu-west-1"
}
```

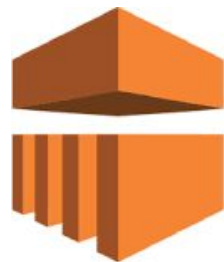


Création de l'environnement Big Data

Choix de la solution technique : EMR

Le service **EMR** permet de louer des serveurs avec des applications préinstallées et configurées

- Facilité de mise en œuvre
- Rapidité de mise en œuvre
- Solutions matérielles et logicielles optimisées



amazon
EMR



TensorFlow



Création de l'environnement Big Data

Configuration de l'EMR

1. Logiciels et étapes : sélectionner les packages dont nous aurons besoin (JupyterHub, TensorFlow, Spark)
2. Modifier les paramètres logiciels : enregistrer et ouvrir les notebooks non plus sur le serveur mais sur S3
3. Sélection du matériel : 1 instance maitre et 2 esclaves
4. Action d'amorçage : installation des packages manquants (ils seront alors disponibles sur toutes les machines)
5. Création d'une paire de clés EC2 : indispensable pour se connecter en SSH



Création de l'environnement Big Data

Connexion à JupyterHub

1. Ouvrir un tunnel SSH vers le nœud maître Amazon EMR

```
mat@LAPTOP-2J2CR7I8:~/aws$ ssh -i ~/oc-cle-2.pem -ND 8157 hadoop@ec2-34-229-9-4.compute-1.amazonaws.com
```

2. Configurer un outil de gestion de proxy : Proxy SwitchyOmega

3. Lancement de JupyterHub

 Logout Control Panel

Files Running Clusters

Select items to perform actions on them. Upload New ▾

<input type="checkbox"/>	0 ▾	 /	Name ▾	Last Modified	File size
<input type="checkbox"/>		 test spark notebook.ipynb			il y a 3 jours



Analyse de code

Preprocessing

```
PATH = 's3://oc-projet8-test'
PATH_Data = PATH+'/sample_data_train'
PATH_Result = PATH+'/Results'
print('PATH:      '+\
      PATH+'\nPATH_Data:  '+\
      PATH_Data+'\nPATH_Result: '+PATH_Result)
```

```
PATH:      s3://oc-projet8-test
PATH_Data: s3://oc-projet8-test/sample_data_train
PATH_Result: s3://oc-projet8-test/Results
```

```
images = spark.read.format("binaryFile") \
    .option("pathGlobFilter", "*.jpg") \
    .option("recursiveFileLookup", "true") \
    .load(PATH_Data)
```

```
images = images.withColumn('label', element_at(split(images['path'], '/'),-2))
print(images.select('path','label').show(5,False))
```

path	label
s3://oc-projet8-test/sample_data_train/apple_crimson_snow_1/r0_40.jpg	apple_crimson_snow_1
s3://oc-projet8-test/sample_data_train/apple_crimson_snow_1/r0_186.jpg	apple_crimson_snow_1
s3://oc-projet8-test/sample_data_train/apple_crimson_snow_1/r0_180.jpg	apple_crimson_snow_1
s3://oc-projet8-test/sample_data_train/apple_crimson_snow_1/r0_190.jpg	apple_crimson_snow_1
s3://oc-projet8-test/sample_data_train/apple_crimson_snow_1/r0_4.jpg	apple_crimson_snow_1

Nous récupérons les images déposées sur S3 en indiquant le chemin associé.

“images” contient alors les informations suivantes : le chemin de l’image dans S3, sa date de modification, sa taille et son contenu en format hexadec

Nous isolons alors la classe de chaque image et créons une nouvelle colonne

```
images.groupBy('label').count().show()
```

label	count
apple_crimson_snow_1	6
apple_6	6
apple_braeburn_1	6
apple_golden_1	6



Analyse de code

Transfert learning

```
model = MobileNetV2(weights='imagenet',  
                    include_top=True,  
                    input_shape=(224, 224, 3))
```

```
new_model = Model(inputs=model.input,  
                  outputs=model.layers[-2].output)
```

```
broadcast_weights = sc.broadcast(new_model.get_weights())
```

```
new_model.set_weights(broadcast_weights.value)
```

le transfert learning consiste à utiliser la connaissance déjà acquise par un modèle entraîné (ici MobileNetV2) pour l'adapter à notre problématique.

Nous allons fournir au modèle nos images, et nous allons récupérer l'avant dernière couche du modèle

Vient ensuite l'étape de broadcast des poids du modèle.

Ce principe consiste à communiquer directement ces poids aux workers.



Réduction du coût de communication



Analyse de code

Transfert learning

```
def preprocess(content):  
    """  
    Preprocesses raw image bytes for prediction.  
    """  
    img = Image.open(io.BytesIO(content)).resize([224, 224])  
    arr = img_to_array(img)  
    return preprocess_input(arr)
```

```
features_df = images.repartition(24).select(col("path"),  
                                           col("label"),  
                                           featurize_udf("content").alias("features")  
                                           )
```

	path	label	features
	s3://oc-projet8-t...	apple_crimson_snow_1	[0.1459774, 0.096...
	s3://oc-projet8-t...	apple_crimson_snow_1	[0.11529742, 0.10...
	s3://oc-projet8-t...	apple_crimson_snow_1	[0.12837058, 0.11...
	s3://oc-projet8-t...	apple_crimson_snow_1	[0.09394203, 0.12...
	s3://oc-projet8-t...	apple_crimson_snow_1	[0.07546184, 0.12...
	s3://oc-projet8-t...	apple_golden_1	[3.6582781E-4, 0....
	s3://oc-projet8-t...	apple_crimson_snow_1	[0.44648364, 0.22...
	s3://oc-projet8-t...	apple_golden_1	[0.012956005, 0.5...

Nous devons redimensionner les images pour qu'elles soient conforme à l'input shape définit

Les features sont alors calculées pour chaque image

Nous obtenons alors un vecteur de taille 1280, ce qui correspond bien à la dernière couche de notre modèle préalablement amputé

```
global_average_pooling2d (Globa (None, 1280)      0      out_relu[0][0]
```




Analyse de code PCA

```
convertUDF = udf(lambda vs: Vectors.dense(vs), VectorUDT())

df_convert = df.select(col("path"), \
    col("label"), \
    convertUDF(col("features")).alias("features") )
```

```
n_components = 5
pca = PCA(
    k = n_components,
    inputCol = 'features',
    outputCol = 'pcaFeatures'
).fit(df_convert)
```

```
df_pca = pca.transform(df_convert)
```

label	features	pcaFeatures
apple_braeburn_1	[0.85709351301193...	[-18.382577847867...
apple_6	[0.77581667900085...	[-2.2894336739582...
apple_crimson_snow_1	[0.44648364186286...	[-11.562520495492...
apple_golden_1	[0.01295600458979...	[6.48727849976521...
apple_crimson_snow_1	[0.07546184211969...	[-11.234026533381...
apple_golden_1	[3.65827814675867...	[6.79889438573909...

Pour que la PCA (présente dans `pyspark.ml.feature`) puisse fonctionner il faut au préalable transformer le type des features en vecteur dense

Nous avons procédé à un simple test pour vérifier que tout fonctionnait :

```
pca.explainedVariance.sum()
```

```
FloatProgress(value=0.0, bar_s
```

```
0.8169154433993073
```

```
pca.explainedVariance.sum()
```

```
DenseVector([0.4453, 0.1849, 0.0886, 0.0611, 0.0371])
```



Analyse de code

Sauvegarde et relecture

On sauvegarde ensuite les résultats sur S3.

```
features_df.write.mode("overwrite").parquet(PATH_Result)
```

La lecture des données se fera ensuite avec

```
df = spark.read.parquet(PATH_Result).toPandas()
```

Démonstration d'exécution du script sur le Cloud

