



PROJET 8:

Déployez un modèle dans le cloud

PROBLEMATIQUE DU PROJET

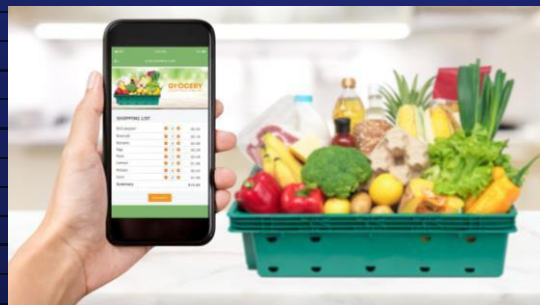


Fruits!

- Fruits! : start-up cherchant des solutions innovantes pour la récolte des fruits.
- Adoption de traitements spécifiques pour chaque espèce de fruits : développement de robots cueilleurs intelligents.



- Objectif : préservation de la biodiversité des fruits.
- Mission à court terme : création d'une application mobile de reconnaissance de fruit par leur photo et affichage d'informations



APPLICATION MOBILE

- Mise en place d'un moteur de classification des images de fruits

01

Traitement des données

- 2 étapes :
- Prétraitements
 - Une réduction de dimension

02

Gestion de l'augmentation rapide du volume des données

Mise en place de
L'architecture Big Data

PLAN D'ETUDE

- I. Présentation des données
- II. Environnement **Big Data**
- III. Le calcul distribué
- IV. Déploiement en local
- V. Déploiement sur aws
- VI. Résultats
- VII. Conclusions

PRESENTATION DES DONNEES

- Images de fruits et leurs labels (catégorie du fruit ou du légume).
- Nombre total d'images : 90483 (131 catégories)



Eggplant



Kaki



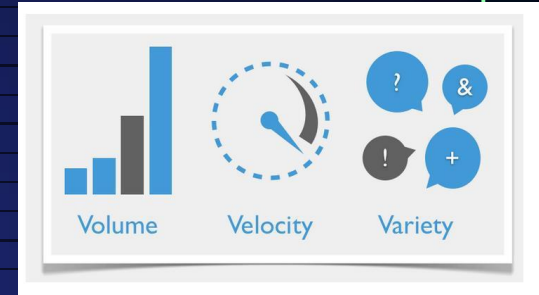
Apple Braeburn

Jeu d'entraînement (dossier Training)	Jeu de test (dossier Test)	Images non rangées par classes (dossier test-multiple_fruits)
<ul style="list-style-type: none">▪ 131 dossiers (catégories)▪ 63 692 images	<ul style="list-style-type: none">▪ 131 dossiers▪ 22 688 images	<ul style="list-style-type: none">▪ 103 images

- Un notebook : chaine de traitements de données réalisées en local puis dans un environnement Big Data AWS EMR.

LES 3 V DU BIG DATA

- **Volume** : quantités astronomiques de données générées
- **Vitesse** : grande vitesse de circulation des fichiers, des données.
- **Variété** : grande variété de données : images, vidéos, musiques, les données de capteurs, les tweets etc.
➡ nécessité de différents traitements

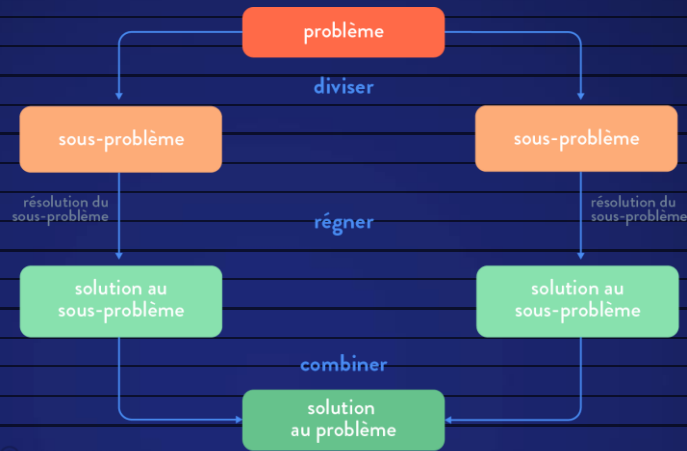


- Les solutions classiques **insuffisantes**
(centraliser le stockage et le traitement des données sur un serveur)
- Exploitation technique du Big Data : **distribution** du stockage et **parallélisation** des traitements sur plusieurs ordinateurs



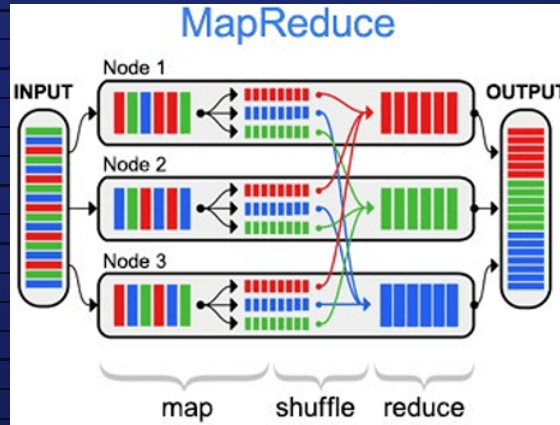
CALCULS DISTRIBUES

- Principe = diviser pour régner.



© scholaparc

- MapReduce :
 - Map = transformation
 - Reduce = agrégation



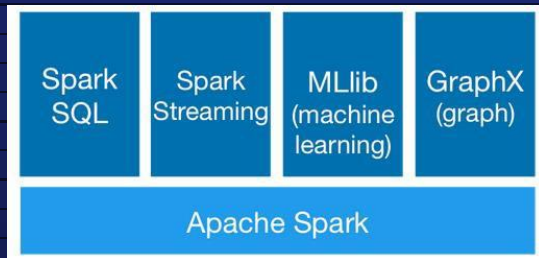
APACHE SPARK



- Cluster (groupe de machines) : mise en commun des ressources de nombreux ordinateurs.
- Prestataires du cloud:



- Spark : **framework de calcul distribué in-memory** pour le traitement et l'analyse des données massives.
(gestion et coordination des taches sur les clusters)
- Nombreuses taches :



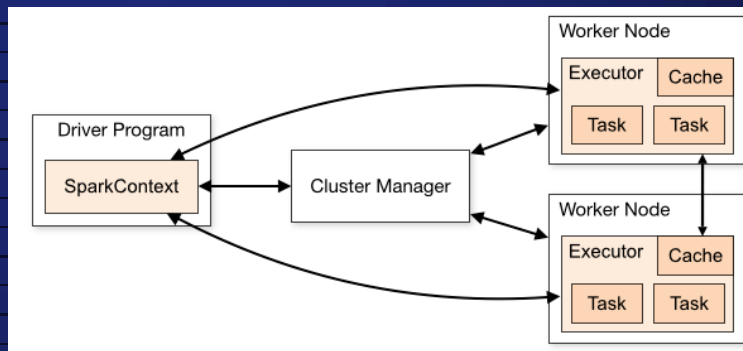
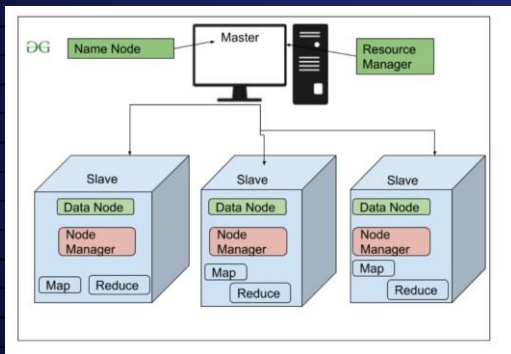
- Utilisation dans le projet de Spark avec Python (**librairie Pyspark**)

APACHE SPARK



- Architecture :

- type maître/esclave.
- Hadoop Map Reduce.
- Distribution des calculs : Spark-session (planification et distribution des taches par le driver entre les executors)



DEPLOIEMENT EN LOCAL

- Objectif : extraction des features des images par la modèle MobileNet (avant dernière couche du modèle).
- Features : vecteurs de dimension (1,1,128).
- Déploiement local avant le cloud : - prototyper les applications en local sur des données faibles avant envois vers un clusters de plusieurs machines pour le traitement de très grandes données.
 - déboguer les applications distribuées en local (plus pratique et moins coûteux).
- En local : parallélisation du calcul par Spark sur les cœurs du processeur.

DEPLOIEMENT SUR AWS

- Configuration du cluster : - choix des logiciels.

Configuration des logiciels

Libérer

<input checked="" type="checkbox"/> Hadoop 3.3.3	<input type="checkbox"/> Zeppelin 0.10.1	<input type="checkbox"/> Livy 0.7.1
<input checked="" type="checkbox"/> JupyterHub 1.4.1	<input type="checkbox"/> Tez 0.10.2	<input type="checkbox"/> Flink 1.15.2
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 2.4.13	<input type="checkbox"/> Pig 0.17.0
<input type="checkbox"/> Hive 3.1.3	<input type="checkbox"/> Presto 0.276	<input type="checkbox"/> ZooKeeper 3.5.10
<input type="checkbox"/> JupyterEnterpriseGateway 2.6.0	<input type="checkbox"/> MXNet 1.9.1	<input type="checkbox"/> Sqoop 1.4.7
<input type="checkbox"/> Hue 4.10.0	<input type="checkbox"/> Phoenix 5.1.2	<input type="checkbox"/> Trino 398
<input type="checkbox"/> Oozie 5.2.1	<input checked="" type="checkbox"/> Spark 3.3.0	<input type="checkbox"/> HCatalog 3.1.3
<input type="checkbox"/> TensorFlow 2.10.0		

- choix des machines

Noeuds de cluster et instances

Choisissez le type d'instance, le nombre d'instances et une option d'achat. [En savoir plus sur les options d'achat d'instances](#)

Les options de la console pour la mise à l'échelle automatique ont été modifiées. [En savoir plus](#)



Type de nœud	Type d'instance	Nombre d'instances	Option d'achat
Maître Groupe d'instances maître - 1	m5.xlarge 4 Cœurs virtuels, 16 Gio de mémoire, stockage EBS uniquement Stockage sur EBS : 64 Gio Ajouter des paramètres de configuration	1 Instances	<input checked="" type="radio"/> A la demande <input type="radio"/> Ponctuelles Utiliser le prix à la demande com
Principal Groupe d'instances principal - 2	m5.xlarge 4 Cœurs virtuels, 16 Gio de mémoire, stockage EBS uniquement Stockage sur EBS : 64 Gio Ajouter des paramètres de configuration	2 Instances	<input checked="" type="radio"/> A la demande <input type="radio"/> Ponctuelles Utiliser le prix à la demande com

DEPLOIEMENT SUR AWS

- choix des packages à installer (création d'un fichier .sh d'instructions d'installations)

▼ Actions d'amorçage

Les actions d'amorçage sont des scripts exécutés lors de la configuration avant le démarrage de Hadoop sur chaque nœud de cluster. Vous pouvez les utiliser pour installer des logiciels supplémentaires et personnaliser vos applications. [En savoir plus](#)

Type d'action d'amorçage	Nom	Emplacement JAR	Arguments facultatifs		
Action personnalisée	Action personnalisée	s3://oc-projet8-data/bootstrap-emr.sh			

Ajouter une action d'amorçage

Sélectionner une action d'amorçage

Configurer et ajouter

Annuler

Précédent

Suivant

```
#!/bin/bash
sudo python3 -m pip install -U setuptools
sudo python3 -m pip install -U pip
sudo python3 -m pip install wheel
sudo python3 -m pip install pillow
sudo python3 -m pip install pandas==1.2.5
sudo python3 -m pip install pyarrow
sudo python3 -m pip install boto3
sudo python3 -m pip install s3fs
sudo python3 -m pip install fsspec
sudo python3 -m pip install tensorflow==2.10.0
```

RESULTATS

- Reduction de dimensions avec 2 composantes principales

Standardisation

```
In [21]: from pyspark.ml.feature import VectorAssembler, StandardScaler, PCA
scaler = StandardScaler(
    inputCol = 'features_vectorized',
    outputCol = 'scaledFeatures',
    withMean = True,
    withStd = True
).fit(df2.select('features_vectorized'))

# when we transform the dataframe, the old
# feature will still remain in it
df_scaled = scaler.transform(df2.select('features_vectorized'))
df_scaled.show(6)
```

features_vectorized	scaledFeatures
[0.65066033601760...	[0.44830321802419...
[0.03623737767338...	[-0.6902513617676...
[0.01539298426359...	[-0.7288770010888...
[0.0, 4.5198950767...	[-0.7574009234000...
[0.0, 4.8245773315...	[-0.7574009234000...
[0.08464313298463...	[-0.6005532252487...

only showing top 6 rows

Application de la PCA

```
In [22]: n_components = 2
pca = PCA(
    k = n_components,
    inputCol = 'scaledFeatures',
    outputCol = 'pcaFeatures'
).fit(df_scaled)

df_pca = pca.transform(df_scaled)
print('Explained Variance Ratio', pca.explainedVariance.toArray())
df_pca.show(5)
```

Explained Variance Ratio [0.07672073 0.05040702]

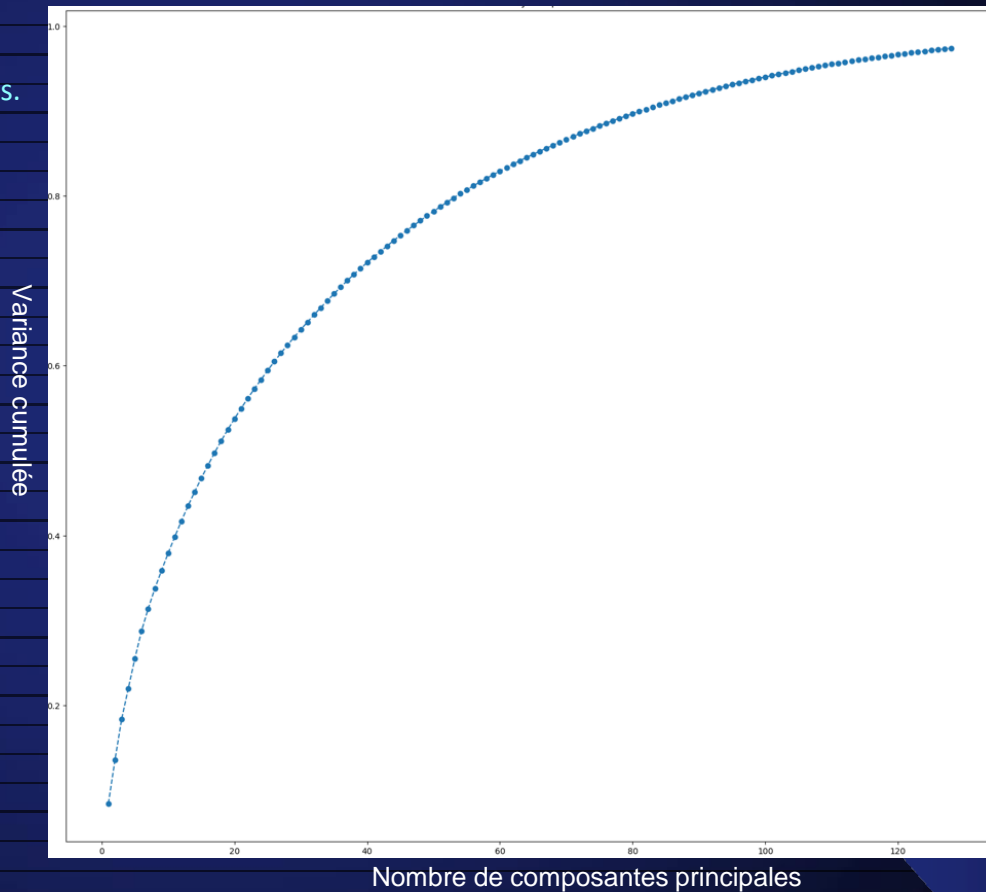
features_vectorized	scaledFeatures	pcaFeatures
[0.65066033601760...	[0.44830321802419...	[-17.287625650450...
[0.03623737767338...	[-0.6902513617676...	[-13.025203309278...
[0.01539298426359...	[-0.7288770010888...	[-9.9118570262535...
[0.0, 4.5198950767...	[-0.7574009234000...	[-12.964916084824...
[0.0, 4.8245773315...	[-0.7574009234000...	[-6.2448371156153...

only showing top 5 rows

- Très faible variance obtenue avec 2 composantes principales : images obtenues conservent très peu de caractéristiques des images d'origine.
- Diminution de la dimension du problème tout en conservant un niveau de variance (informations) acceptable.

RESULTATS

- Variance expliquée en fonction du nombre de composantes principales (variance obtenue en local pour un nombre limité de données (300 images)).
- Variance > 80 % pour 80 composantes principales.



CONCLUSIONS :

- Réalisation du déploiement du modèle dans le cloud sur des grandes données (extraction des features et réduction de dimension).
- Réduction de dimension : besoin de déterminer le minimum de composantes principales tout en perdant moins d'informations possibles.
- Améliorations possibles: choix d'un autre modèle, d'une autre architecture big data...
- ➡ meilleur compromis entre meilleurs choix (résultats acceptables) et les plus faibles coûts possibles.

