01

10 juillet 2020

OPENCLASSROOMS

Soutenance projet 8

Déployer un modèle dans le cloud

Evaluateur : Julien Heiduk

Etudiant : Louis Birenholz

02

Presentation Highlights

- **Introduction**
- **Présentation des outils utilisés**
  1. WSL2
  2. Spark
  3. Amazon Web Services (EC2 & S3)
- **Preprocessing**
  1. Extraction des features
  2. Réduction dimensionnelle
  3. Stockage sur S3
- **Conclusion et recommandations**

03

Introduction

Fruits!

- **Background du projet:** **“Fruits!”,** une jeune start up de l’agritech. Recherche des solutions innovantes pour la récolte des fruits.
- **Objectifs:** Développer dans un environnement **Big Data** une première chaîne de traitement des données.
- **Data:** 90483 images / Dimension : 100x100 pixels / 131 (fruits & légumes)

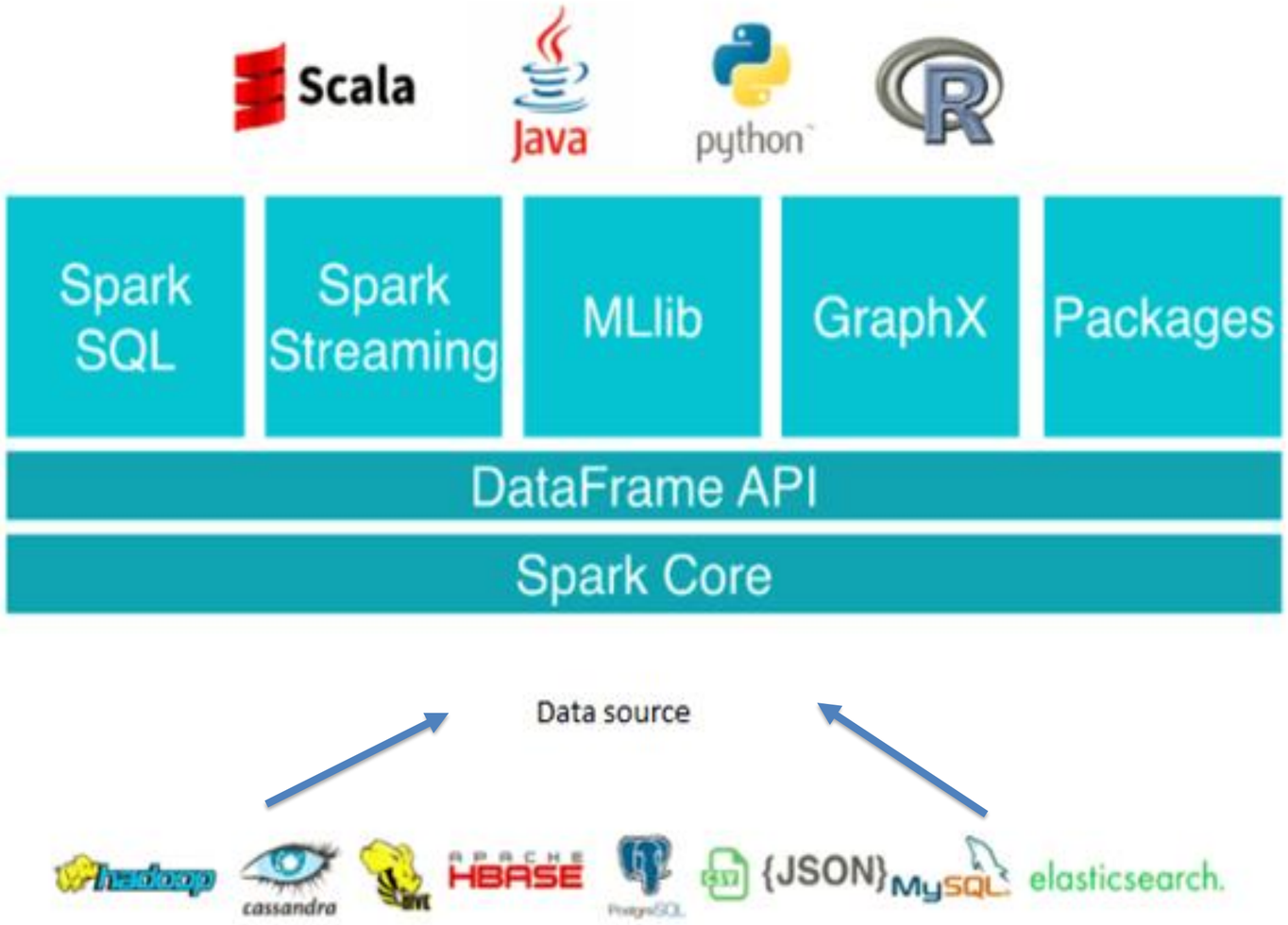
Outils utilisés

- **WSL2 & Ubuntu :** Windows Subsystem for Linux est une couche de compatibilité permettant d'exécuter des exécutables binaires Linux de manière native sur Windows 10 et Windows Server 2019.

- **Spark :** Spark est un *framework* open source de calcul distribué. Il s'agit d'un ensemble d'outils et de composants logiciels structurés selon une architecture définie.

  - Moteur de traitement parallèle de données **open source.**
  - Traitement **In-memory (Stockage intermédiaire en RAM).**
  - **100 fois** plus rapidement que Hadoop MapReduce in-memory
  - Grande quantité de bibliothèques d'algorithmes (SparkML, Spark Streaming…)
  - Très largement adopté par les gestionnaires de **datalake.**
  - **Languages :** Python/Java/R/Scala …
  - La lazy evaluation.

Spark


Scala
Java
python
R
Spark SQL
Spark Streaming
MLlib
GraphX
Packages
DataFrame API
Spark Core
Data source
hadoop
cassandra
HBASE
{JSON}
MySQL
elasticsearch.

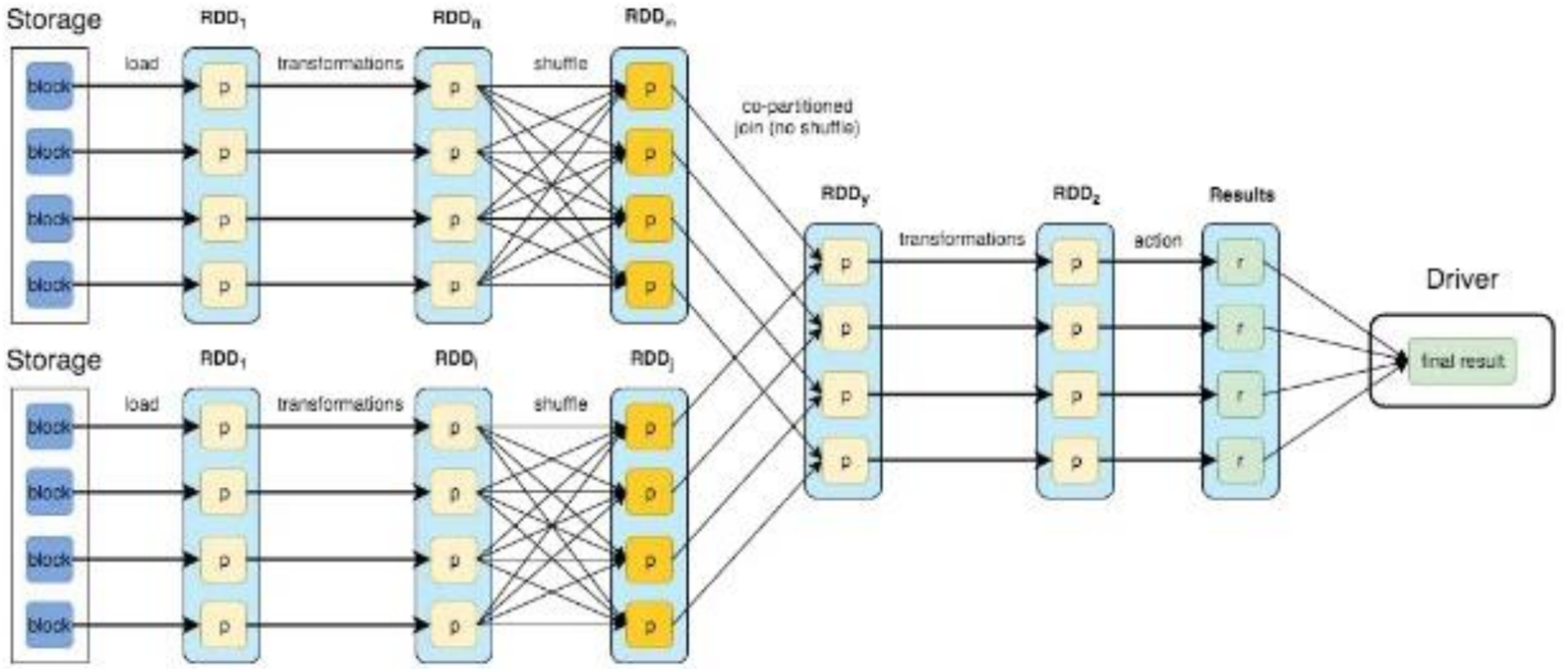
RDD, Transformation & Action

RDD : Resilient Distributed Dataset

- **Dataset** : Il s'agit d'un jeu de données qui se parcourt comme une collection.
- **Distributed** : Cette structure est distribuée afin d'être découpée pour être traitée dans les différents nœuds.
- **Resilient** : Il est résilient, car il pourra être relu en cas de problème.

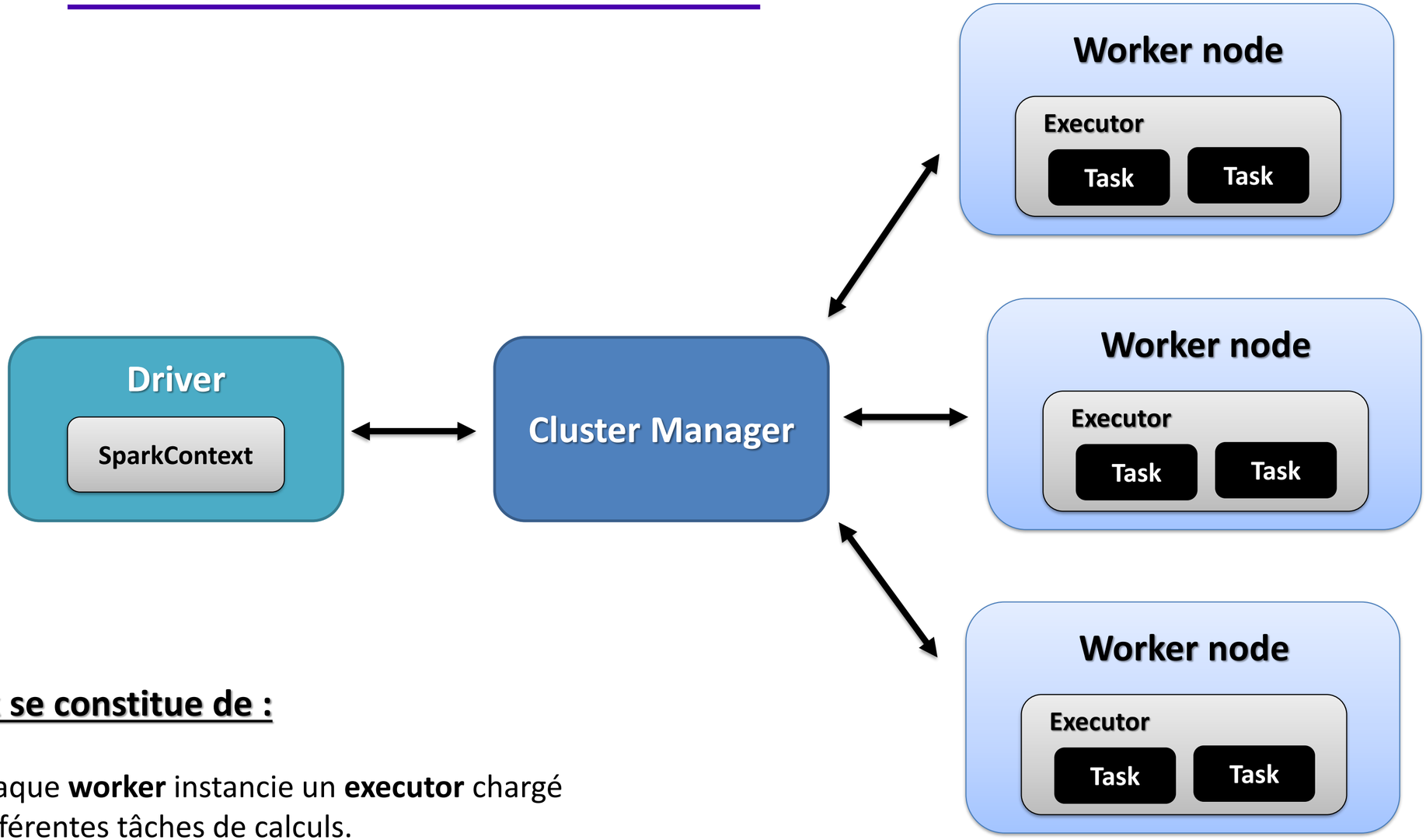
Operations sur les RDD :

- **Les Transformations :** filter(), union() …
- **Les Actions :** count(), first(), show() …



Exemple de Directed Acyclic Graph (DAG)

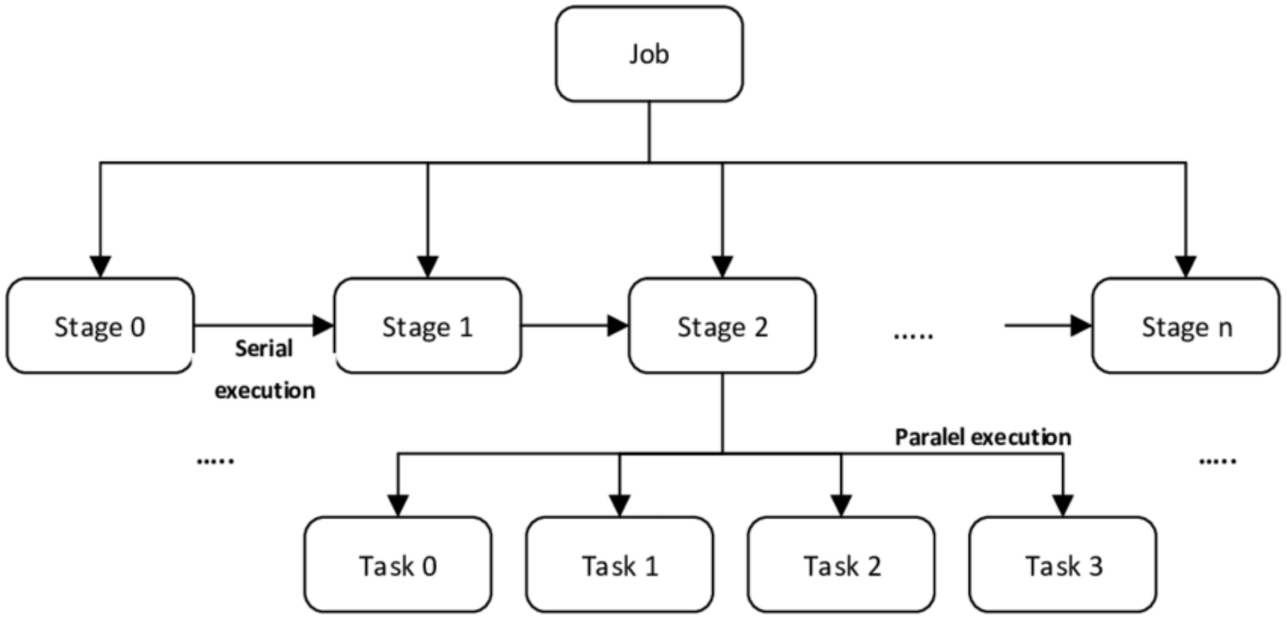
Architecture

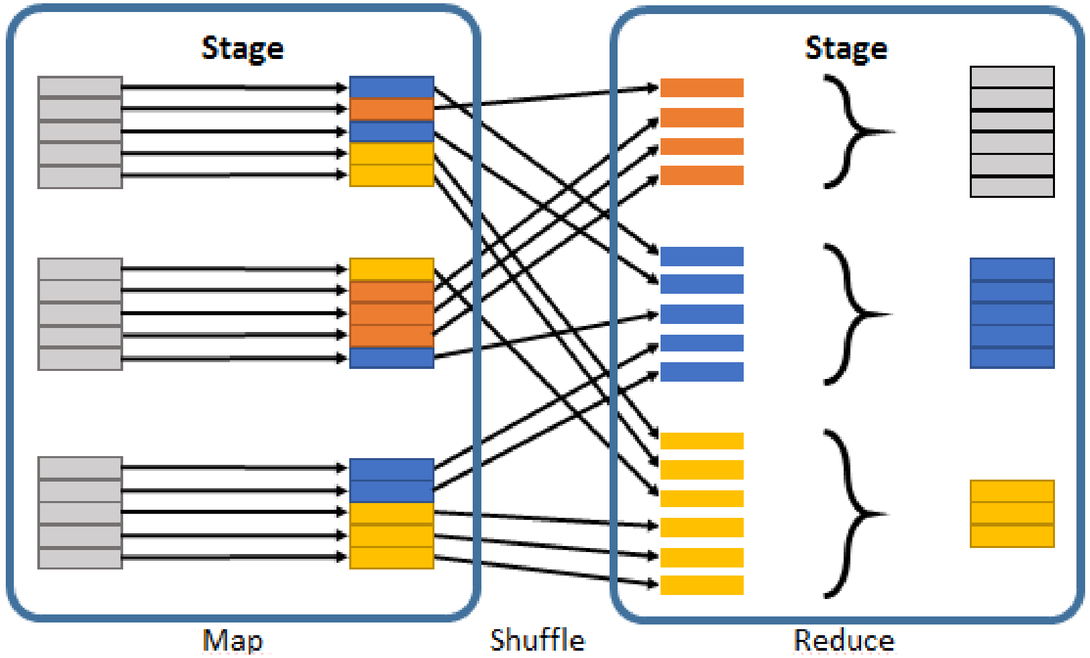


Un cluster Spark se constitue de :

- Des **workers** : chaque **worker** instancie un **executor** chargé d'exécuter les différentes tâches de calculs.
- Un **driver** : chargé de répartir les tâches sur les différents **executors**.
- Un **cluster manager** : chargé d'instancier les différents **workers**.

Job, Stage and Task





- Une application **Spark** est constitué d'un ensemble de **Job.**
- Un **Job** est constitué d'un ensemble de **Stage**
- Un **Stage** est constitué d'un ensemble de **Task** se terminant par un **Shuffle**

Environnement Big Data sur le Cloud

Amazon S3

Lecture des images depuis le bucket S3

Ecriture des données après preprocessing

Amazon EC2

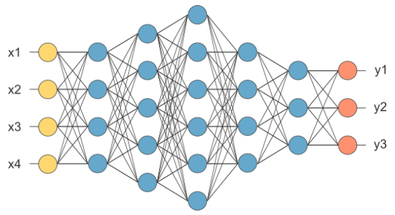
- **Stockage des données dans un Bucket S3**
- **Path des images :** s3a://monbucketS3/Training/image1.jpg
- **Région :** us-east-1

- Utilisation de **JupyterLab** avec Spark
- **Type d'instance EC2 :** t3.large
- Utilisation de la librairie SparkDL
- Calcule distribué sur : X cœurs
- **Région :** us-east-2

Type d'instance	Zone de disponib
t2.micro	us-east-2b
t3.large	us-east-2b

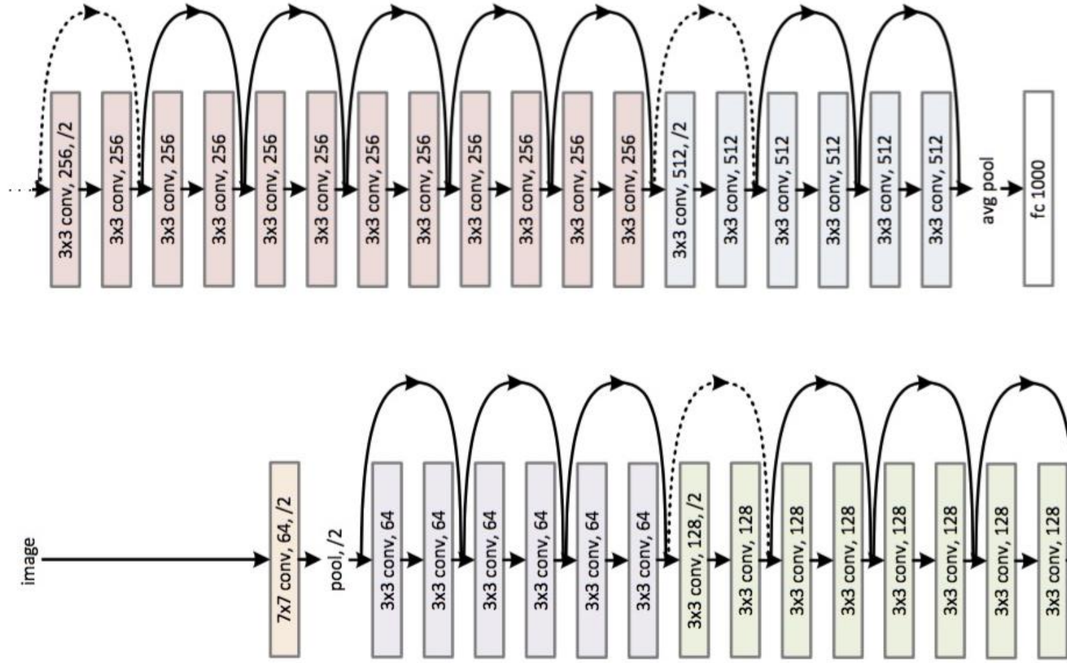
Spark DL :

+



- Package venant de **Data Bricks**
- Image loading
- Applying pre-trained models as transformers in a Spark ML pipeline
- Transfer Learning & Feature Extraction
- Distributed hyperparameter tuning
- Deploying models in DataFrames and SQL

ResNet50 :



Preprocessing des images

- **Loading from S3 :**

```
path = "s3a://projet8louis/Training_reduced"

# Read data in a DataFrame.
df_training = ImageSchema.readImages(path, recursive=True)
```

- **Extracting features with transfer learning on ResNet50 :**

```
# Feature extraction with ResNet50.
featurizer = DeepImageFeaturizer(inputCol="image", outputCol="features", modelName="ResNet50")
```

- **Réduction dimensionelle (ACP) :**

```
# PCA.
pca = PCA(k=2048, inputCol="features", outputCol="pca")
model = pca.fit(df_extract)

# How many components are necessary to explain 95% of the inertia ?
for c,i in enumerate(np.cumsum(model.explainedVariance)):
    if i>=0.95:
        print('{} composantes principales sont nécessaires'.format(c))
        break
```

```
Il faut 12 composantes principales
```

- **Extraction des labels dans les paths :**

```
# Création de la udf.
split_path_udf = udf(lambda z: split_path(z), StringType())

# Dataframe with principal component & label.
df_final = df_pca.select('image.origin','pca', split_path_udf('image.origin').alias('label'))
```

Stockage des données après preprocessing

origin	pca	label
s3a://projet8loui...	[-16.262893200262...	Apple Braeburn
s3a://projet8loui...	[-16.546459805582...	Apple Braeburn
s3a://projet8loui...	[-16.402209911536...	Apple Braeburn
s3a://projet8loui...	[-16.892880706972...	Apple Braeburn
s3a://projet8loui...	[-16.834413596396...	Apple Braeburn
s3a://projet8loui...	[-16.408049713471...	Apple Braeburn
s3a://projet8loui...	[-16.087327103765...	Apple Braeburn
s3a://projet8loui...	[-17.259288461497...	Apple Braeburn
s3a://projet8loui...	[-16.678418207681...	Apple Braeburn
s3a://projet8loui...	[-16.395502838567...	Apple Braeburn
s3a://projet8loui...	[-16.640781726634...	Apple Braeburn
s3a://projet8loui...	[-14.619737210615...	Apple Braeburn
s3a://projet8loui...	[-15.973752392968...	Apple Braeburn
s3a://projet8loui...	[-16.322170235248...	Apple Braeburn
s3a://projet8loui...	[-16.271037508856...	Apple Braeburn
s3a://projet8loui...	[-13.140675415817...	Apple Red 1
s3a://projet8loui...	[-15.289659248662...	Apple Red 1
s3a://projet8loui...	[-15.266879077533...	Apple Red 1
s3a://projet8loui...	[-15.114300403045...	Apple Red 1
s3a://projet8loui...	[-15.910337996484...	Apple Red 1

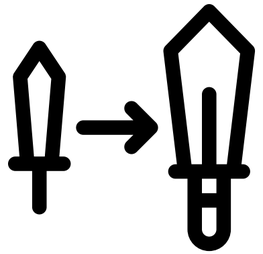
only showing top 20 rows

Stockage au format .parquet

Amazon S3

Conclusion et recommandations

- **Utilisation de type d'instance EC2 + performantes suivant l'utilisation de l'entreprise**
  - Usage général
  - Mémoire optimisée
  - Calcule accéléré
  - Stockage optimisée



- **Utilisation de Amazon EMR**
  - → Cluster computing
  - Infrastructure non rigide
  - Coûts avantageux
  - Découplage calcul/stockage
  - Passage à l'échelle

- **Optimisation des coûts en fonction de l'emplacement des serveurs**
  - Prix
  - Accessibilité
  - Panne

Merci de votre attention