

05/02/2021

P8 DataScientist – OpenClassrooms

Etienne Lardeur

Mentor : **Xavier Tizon**

Evaluateur : **Julien Heiduck**



Fruits!

Solutions innovantes pour la récolte des fruits Le robot cueilleur intelligent

*Développer dans un environnement Big Data, une chaîne de traitement d'images incluant **preprocessing** et **réduction de dimension***

Sommaire

- Introduction 5'
 - Contexte, jeu de données, use case -> **slides 3, 4 et 5**
- Dispositif proposé et rôle de chaque brique 8'
 - Architecture: base, dispositifs en Local ou Cloud -> **slides 6 et 7**
 - Illustration Spark UI -> **slides 8**
- Chaînes de traitement 7'
 - Prototype, transposition Cloud, complémentarité -> **slides 9 et 10**
- Conclusion, Recommandations 5' -> **slides 11 et 12**
- Question-réponses

https://github.com/EtienneLardeur/P8_FruitsRecognition

Contexte

○ Finalité : robots cueilleurs intelligents

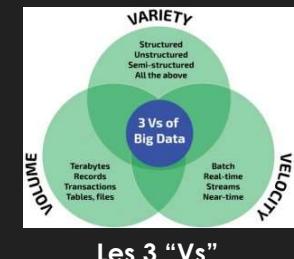


- **Antériorité** (projets Magali (1985) et Citrus (1990)) :
 - Enjeux : remédier à la pénurie de main d'œuvre (**Saisonnalité & Savoir-faire** ✕ **Productivité**) [1]
 - Principaux verrous progressivement levés : performance caméras, **puissance de calcul** et avènement du GPS.
- **Actualité** : investissements R&D, enjeux de **Propriété Intellectuelle**
 - Tevel aerobotics (Isrl), Robocrops (UK), Octinion (Blg), Airsprid (Fr), ...

○ Première étape : populariser via une App mobile de reconnaissance de fruits



- **Justifie** une architecture Big Data (modèle des 3 « V » [2])
 - **Volume** : f(données labellisées, variétés, stades de développement, nouvelles données)
 - initial : **1Mo/i**, pre-process : **10ko/i**, soit proto en **Go** et usage en **PétaOctet**
 - **Vitesse** : collecte et partage de données, puissance de calcul, latence à minimiser.
 - **Variété** (sources et structures de données) :
 - environnement : imagerie, géolocalisation, capteurs...
 - bases de données tierces : ex, bio-agresseurs et auxiliaires [3]



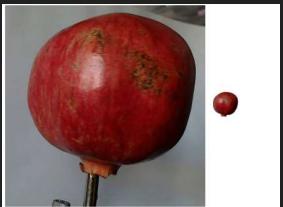
○ Et voici comment le **Big Data** investit le « champ » de l'**Arboriculture** 😎

3

- [1] itw P. Grenier, Technique de l'Ingénieur, 09/2020
[2] Big Data – University of Granada (research group)
[3] Bases de données ecophytopic.fr

Jeu de données

- **Dataset Kaggle [1] riche de 131 variétés de fruit et légumes - labellisées**



- Photos "360°" extraites d'une captation spécifique :
 - rotation tri-axiale
 - post-traitée (fond blanc reconstruit + resizing 100x100 pixel)
 - Train : 67 692 Fichiers / Test : 22 688 Fichiers

- **Intérêt : focus sur feature extraction et stratégies de classification**

- **Limites : procédé initial lourd et non représentatif**

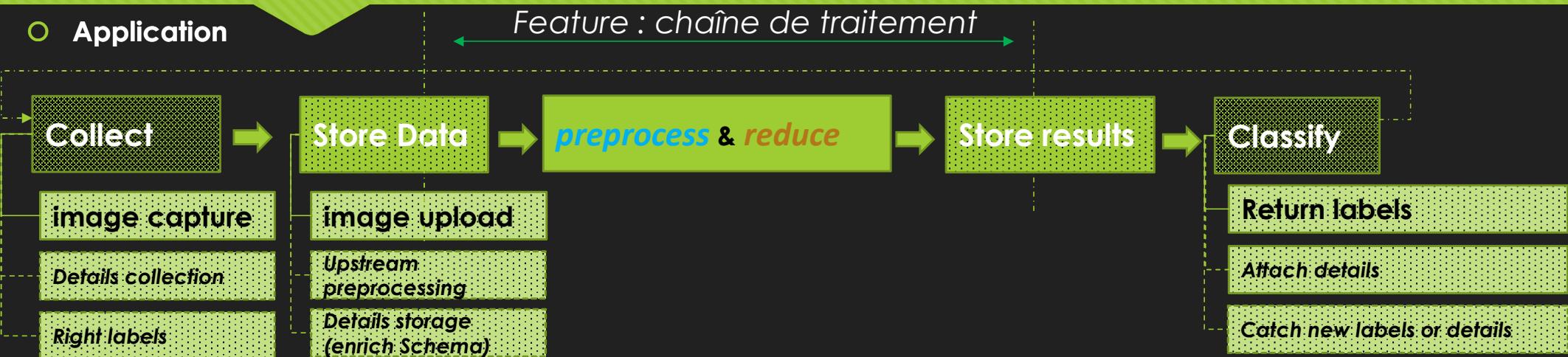
- Diversités d'aspects, formes et couleurs (croissance et maturité)
 - Implique l'ajout d'un preprocessing 'conditions réelles' : cropping & background-removal, resizing

- **Opportunité : enrichissement des données en conditions réelles, partiellement labellisées**

4

Use case

- Application



- Rôles / compétences



- Métriques : disponibilité, rapidité, précision, ...
- Enjeux : parallélisation, resilience, évolutivité - scalabilité
- Contexte : landscape Data & AI [1] émergence DataOps [2],



[1] FirstMark : 2020/09/2020-Data-and-AI-Landscape

[2] exemple positionnement de la société Saagie

Use case

○ Application

Feature : chaîne de traitement

Collect

image capture

Details collection

Right labels

Store Data

image upload

Upstream
preprocessing

Details storage
(enriched)

preprocess & reduce



Store results

Classify

Return labels

Attach details

Catch new labels or details

○ Rôles / compétences



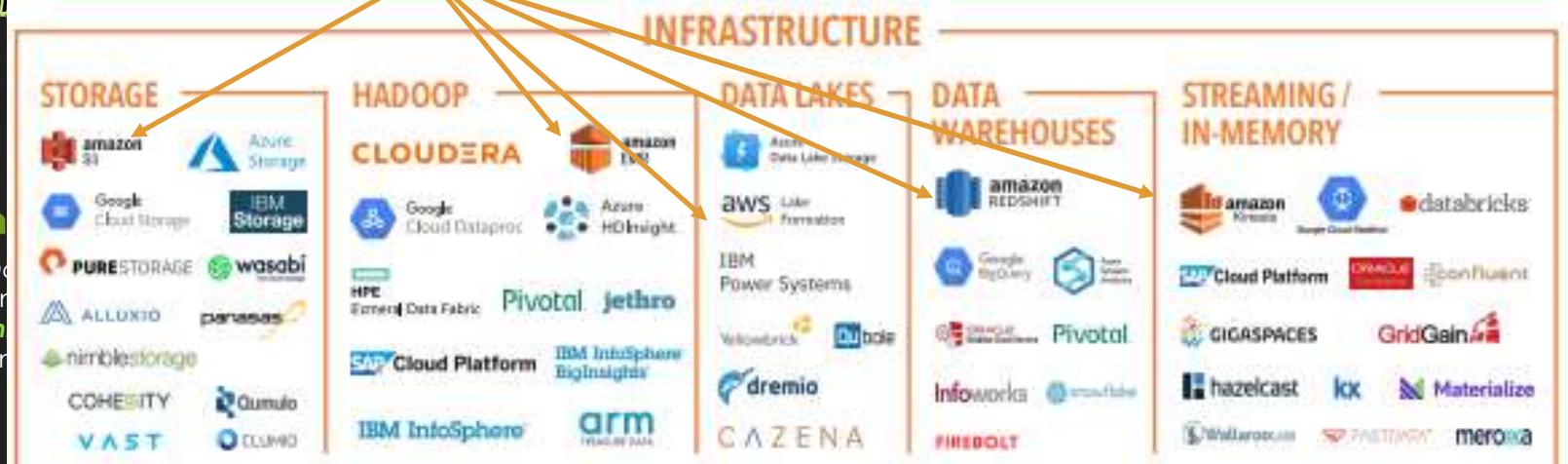
Team
"Business"



Team Data
... Enginee
... Archi
... Scien



Team IT



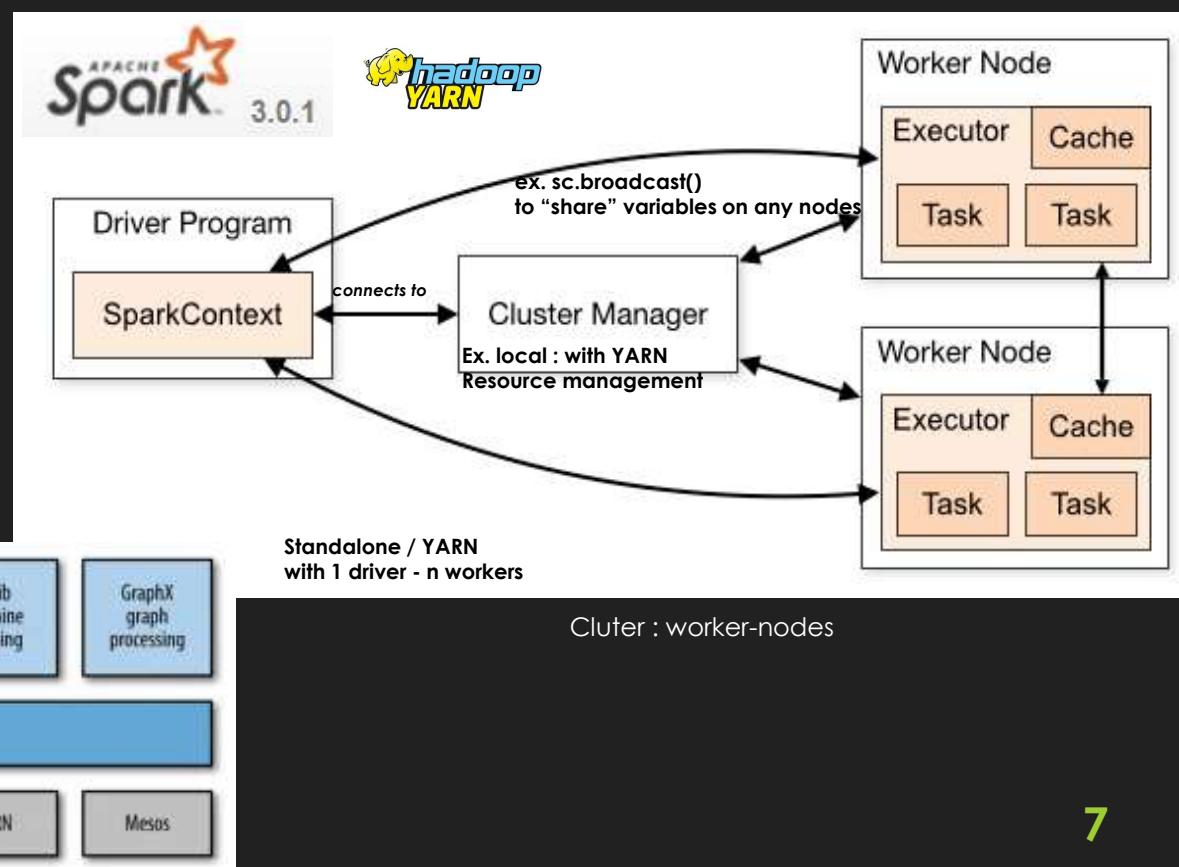
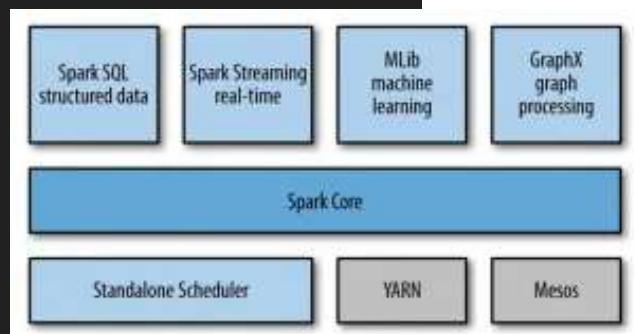
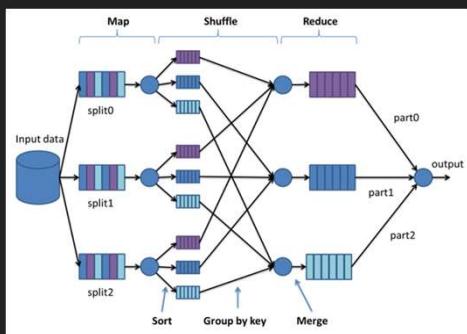
6

[1] FirstMark : 2020/09/2020-Data-and-AI-Landscape

[2] exemple positionnement de la société Saagie

Architecture, base

- Fondements : ecosystem Hadoop + framework Spark
 - Base Hadoop map/reduce + traitement “in memory”
 - Base Resilient Distributed Datasets + Spark DataFrame
 - Assurant la **Parallélisation** des opérations
 - **Résilient** au moyen de graphes acycliques orientés (d'où la tolérance au pannes)
 - “Lazy evaluation” (Transformation vs Action)
 - Performance conditionnée par la stratégie de partitioning [1]

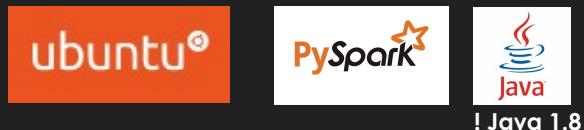


[1] Medium: On Spark Performance and partitioning strategies

Architecture, dispositifs

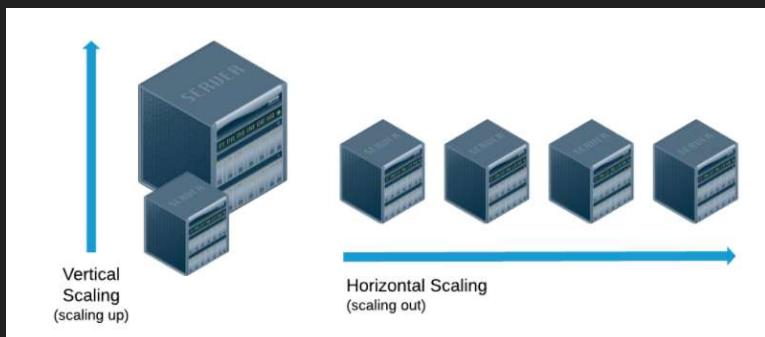
○ Dispositif Local (prototype) : *montée en compétence ok*

- script pySpark sur linux : ubuntu WSL (yc "tunnelisation" ssh)



○ Dispositifs Cloud possible pour assurer la scalabilité : *montée en compétence en cours*

○ Scaling Vertical versus Horizontal [1]



+ L'écosystème AWS:

- "Servitudes" d'architecture :
 - Solutions de stockage (S3) + Gestion de permissions (IAM)
- Alternatives calcul :
 - Puissance de calcul (dimension "fixe") (EC2)
 - Cluster management – logique Spark workers-nodes (EMR- n EC2s)
 - Environnement de développement (ex. Jupyter Notebooks)
- Et au delà...

+ autres solutions **PAAS**...

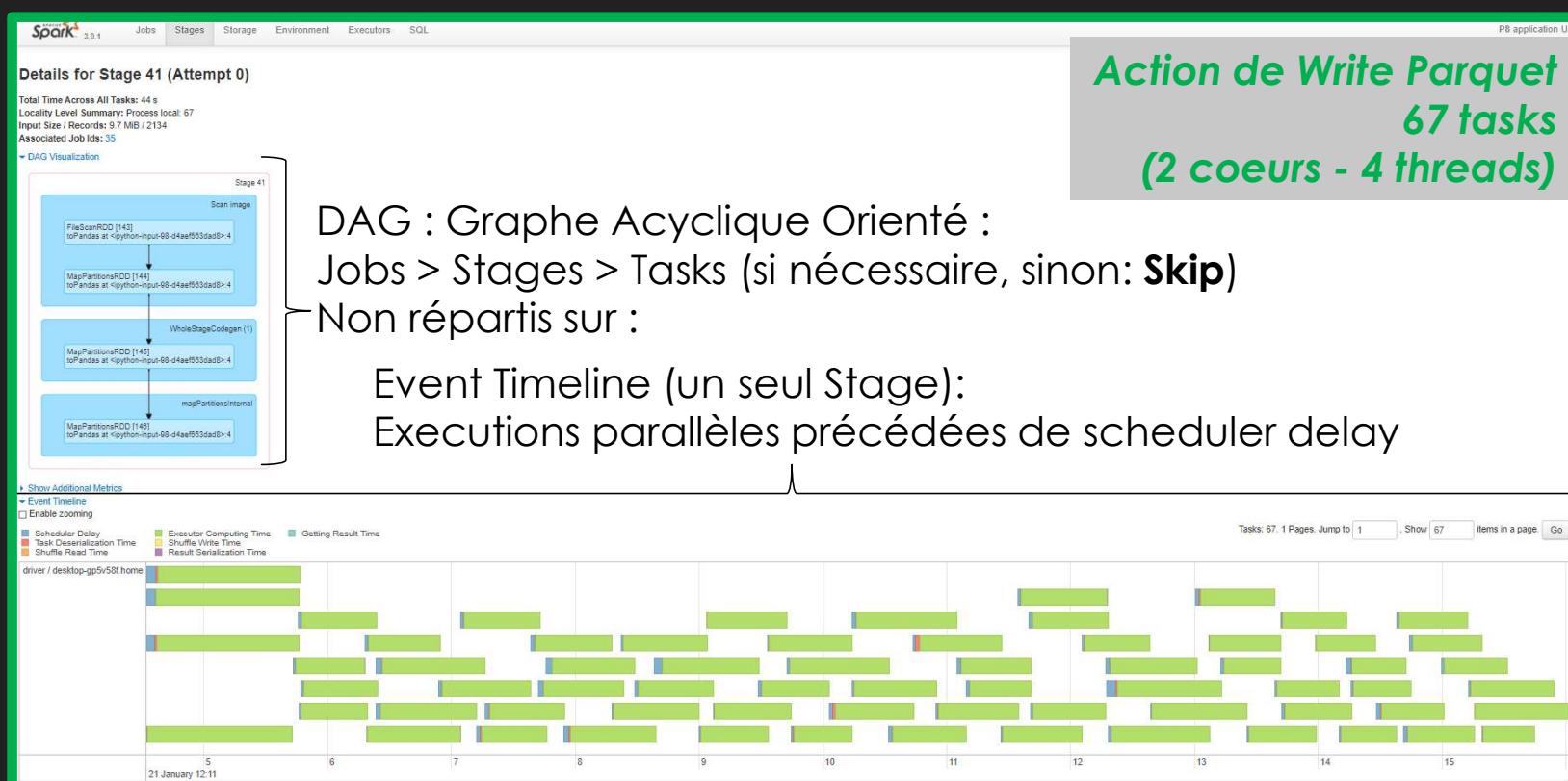
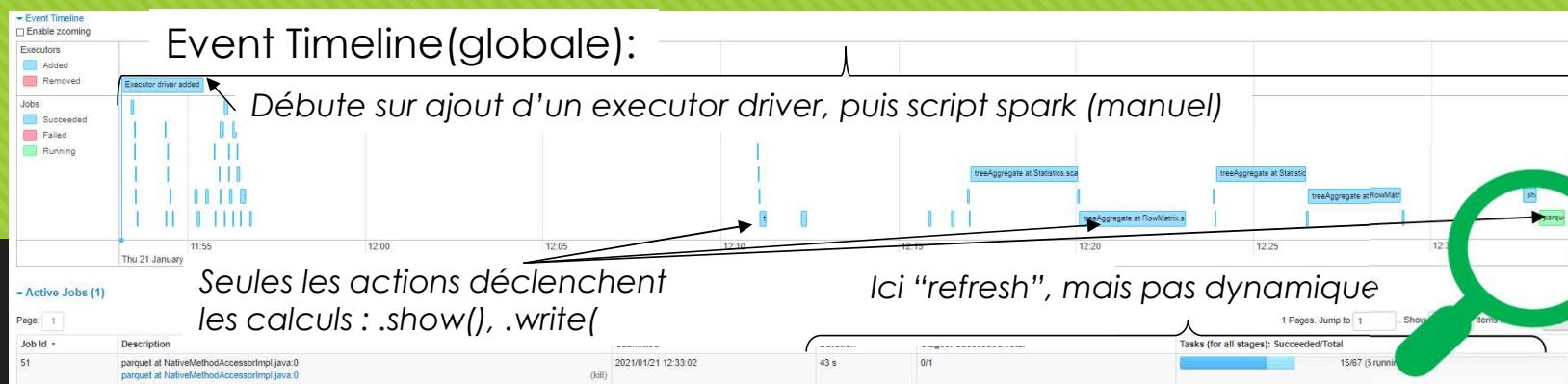


[1] *Scaling Horizontally vs. Scaling Vertically*



Spark UI

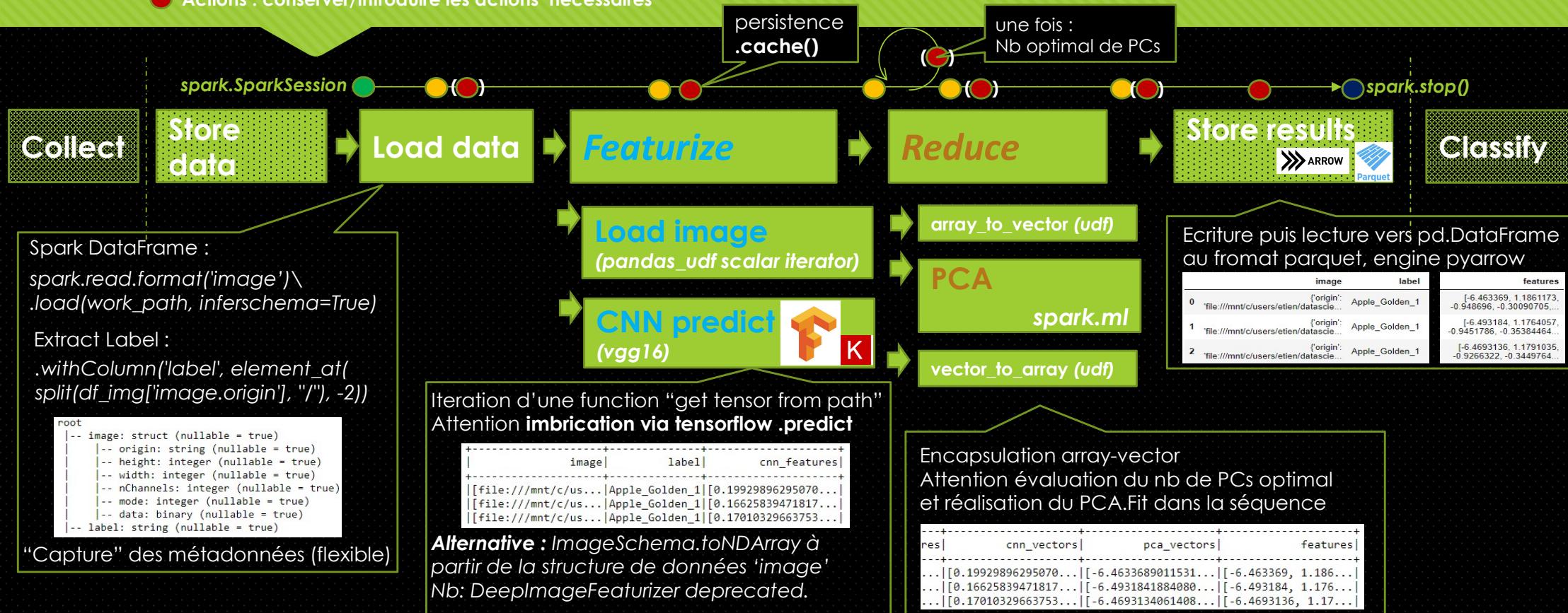
- O **Outil de monitoring** disponible sur un port en complément
- O **Enregistre** le log (py4j)
- O Instructif mais empilage des technologies mises en oeuvre **déroutant**



Chaine de traitement, pySpark local

● Transformations : non évaluées

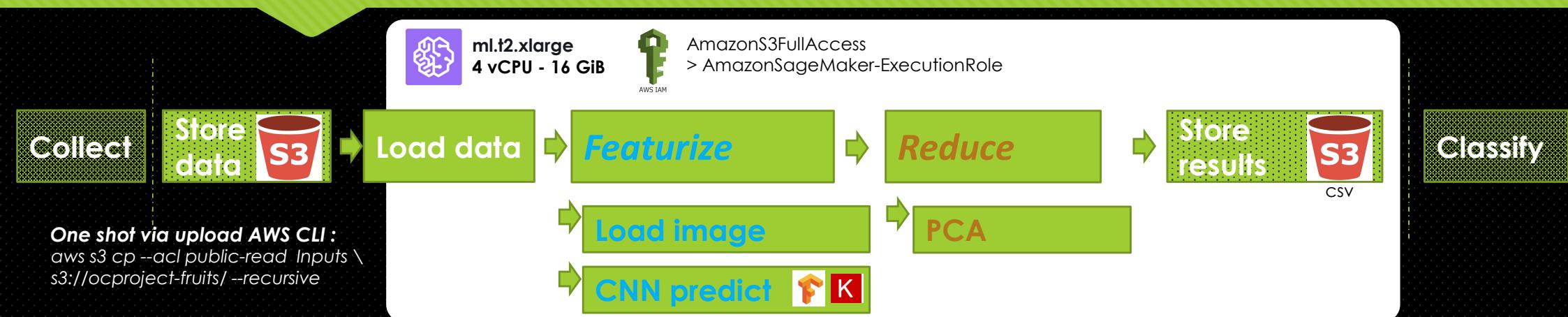
● Actions : conserver/introduire les actions nécessaires



- NOMBREUSES variantes de séquences possibles pour un même but

- Types et Schemas des données, packages exploités (! versions, maintenance), codage optimal, ...

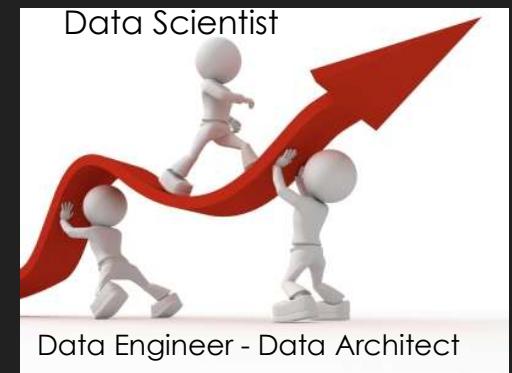
Chaîne de traitement Cloud



- Gain de performance initial **Spark Local** : **Load** (image) + **Store** (parquet) **impressionnant**
- Fonctionnalité “Spark Cloud” **valide** (**volume**, **variété**, **vitesse**):
 - Fonctionnalité et performance **Load** et **Store dégradée** (disponibilité packages - versions et faisabilité)
 - Meilleure performance **Featurize** (tensorflow exploite la puissance de calcul, optimisable à état de l'art [1] + “pruning” cnn)
 - Gains substantiels **Reduce**
- **Poursuivre** linstanciation du meilleur assemblage
- Explorer le **Streaming** et recherche de réduction « incrémentale » [2]

Conclusion

- Un feature au sein du projet, au sein de l'approche business de l'entreprise, au sein d'une mutation digitale
- Sensibilisation aux compétences expertes requises:
 - Projets **collaboratifs** aux mains **d'équipes** pluridisciplinaires !



Recommendations

Perspectives techniques (industrialisation)

- Etat de l'art choix technologiques:
 - Transfert learning avec fine tuning pour meilleur accuracy
 - Feature map pruning pour simplification et rapidité
- Code refactoring (selon la technologie : langage **Scala**)
- Scaling vertical vs horizontal: **analyse technico-économique**
- Extension cas réels logique utilisateur (preprocessing)
- Exploitation pour le développement du Robot Cueilleur...



Perspectives orientées business :

- Contractuel & économie de la solution choisie
- Exploiter l'application pour enrichir en dynamique :
 - Labellisation, informations additionnels
 - Utilisateurs ou professionnels
- Augmenter le cycle de vie
 - Cueillette et l'entretien : maturité, pathologies, conseils de taille



Partenariats – collaboration - compétences

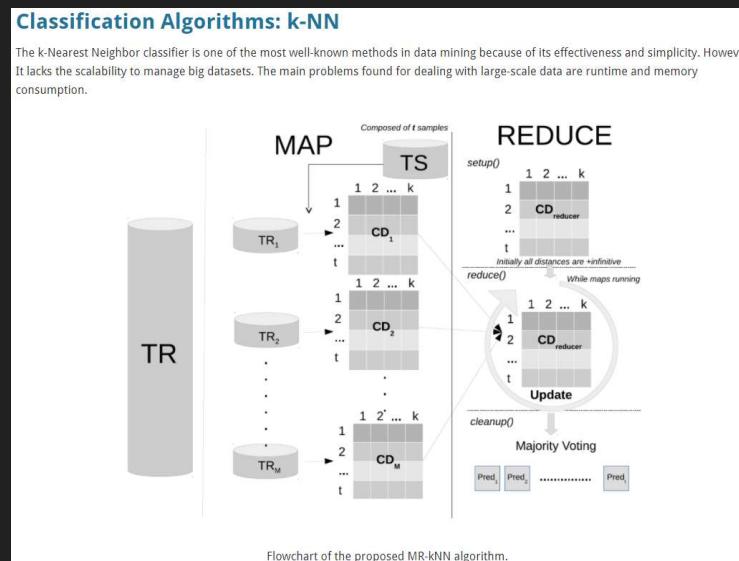
- Accélérer la recherche (ex. partenaire universitaire)
- Optimiser la mise en oeuvre (ex. collab. acteur majeur)
- Recours à l'expertise & montée en compétence
 - Ex. containerization, ML Ops, ...

Questions / réponses

Merci de votre attention !

Annexe : parallélisation

- En pratique le framework Spark lui même implique de s'impliquer dans le détail de chaque calcul afin de construire l'algorithme le plus efficient
- Exemple de proposition k-NN :



Sous-section du code, steps intermédiaires (fails)

-- Original upload of images into storage Bucket --

```
# bulk rename to remove spaces out of folders name
# rational : spark.read.format("image").load(path) requires no space in path
# warning : do not apply until checking the parent location
def rename_folders(parent):
    for path, folders, _ in os.walk(parent):
        for i in range(len(folders)):
            new_name = folders[i].replace(' ', '_')
            os.rename(os.path.join(path, folders[i]), os.path.join(path, new_name))
            folders[i] = new_name
# warning : only one time
rename_folders('Inputs/Training')
# then upload S3 through awscli : aws s3 cp Inputs s3://oc-p8-fruits-storage/ --recursive
```

-- OOM Java heap space failure demonstration & check size

OOM occurs over a given size of vectors matrix ($2800 < \text{nb_col} < 3000$, even with tiny nb_row).

Repartition increase computational time with no substantial effect on OOM issue.

```
# build the Spark DataFrame as vectors matrix
def test_vec_df(nb_row, nb_col, nb_repartition):
    pdf = pd.DataFrame(np.random.rand(nb_row, nb_col))
    df = spark.createDataFrame(pdf)
    input_cols = df.columns
    df = df.repartition(nb_repartition)
    assembler = VectorAssembler(
        inputCols=input_cols,
        outputCol='features')
    df_vec = assembler.transform(df)
    return df_vec.select('features')
```

-- configuration logicielle

<https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-aws/2.7.7>

Compile Dependencies (4)			
Category/License	Group / Artifact	Version	Updates
Cloud Computing Apache 2.0	com.amazonaws > aws-java-sdk	1.7.4	1.11.943
JSON Lib Apache 2.0	com.fasterxml.jackson.core > jackson-databind	2.2.3	2.12.1
JSON Lib Apache 2.0	com.fasterxml.jackson.core > jackson-annotations	2.2.3	2.12.1
Apache 2.0	org.apache.hadoop > hadoop-common	2.7.7	3.3.0

```
"" os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages
com.amazonaws:aws-java-sdk:1.7.4,\
org.apache.hadoop:hadoop-aws:2.7.3,\
com.amazonaws:aws-java-sdk-s3:1.11.762 pyspark-shell'''
```

```
# create input & try to reduce with PCA
test_df = test_vec_df(2200, 512, 4)
pca = PCA(k=16, inputCol="features", outputCol="pcafeatures")
model = pca.fit(test_df)
results_df = model.transform(test_df)
results_df.show(5, True)
```

Sous-section du code, steps intermédiaires (fails)

Notebook EMR:
Instructions erronées



Étape 1 : Ouvrez un tunnel SSH vers le nœud maître Amazon EMR - [En savoir plus](#)

Windows **Mac/Linux**

1. Ouvrez une fenêtre de terminal. Sur Mac OS X, choisissez Applications > Utilities > Terminal. Sur d'autres distributions Linux se trouve généralement sur > Accessories > Terminal.
2. Pour établir un tunnel SSH avec le nœud maître à l'aide d'un transfert de port dynamique, tapez la commande suivante. Remplacez ~~/AWS_KeyOC.pem par l'emplacement et le nom du fichier de clés privées (.pem) utilisé pour lancer le cluster.
`ssh -i ~~/AWS_KeyOC.pem -ND 8157 hadoop@ec2-34-249-246-31.eu-west-1.compute.amazonaws.com`

Remarque : le port 8157 utilisé dans la commande est un port local inutilisé qui a été sélectionné de façon aléatoire.

3. Cliquez sur Yes pour ignorer l'avertissement de sécurité.

```
etienne@DESKTOP-GP5V58F:~/ssh$ ssh -i AWS_KeyOC.pem -ND 8157 hadoop@ec2-34-249-246-31.eu-west-1.compute.amazonaws.com
ssh: connect to host ec2-34-249-246-31.eu-west-1.compute.amazonaws.com port 22: Resource temporarily unavailable
etienne@DESKTOP-GP5V58F:~/ssh$ ssh -i 'AWS_KeyOC.pem' -L 5511:127.0.0.1:8888 hadoop@ec2-34-249-246-31.eu-west-1.compute.amazonaws.com
ssh: connect to host ec2-34-249-246-31.eu-west-1.compute.amazonaws.com port 22: Resource temporarily unavailable
etienne@DESKTOP-GP5V58F:~/ssh$
```

Sous-section du code, steps intermédiaires (fails)



- Selection d'une AMI : d'ubuntu vierge à BigDL
- Choix d'une capacité d'instance : EC2 t2.xlarge
- IAM Role défini pour la connection à S3 : IAMRoleEC2toS3
- l'instance est accessible de façon 'classique' via SSH,
ssh -i 'ocoproject.pem' -L 5511:127.0.0.1:8888 ubuntu@ ec2-15-237-107-48.eu-west-3.compute.amazonaws.com
- sudo apt update
- sudo apt install python3-pip
- sudo pip install jupyter
- Ensuite la commande jupyter notebook, lance un kernel accessible depuis notre machine à l'url http://127.0.0.1:5511/
- Copie de scp -i ~/ssh/ocoproject.pem spark-3.0.1-bin-hadoop2.7.tgz ubuntu@ec2-15-237-107-48.eu-west-3.compute.amazonaws.com:/home/ubuntu
- tar -zxf spark-3.0.1-bin-hadoop2.7.tgz
- sudo mv spark-3.0.1-bin-hadoop2.7 /opt/spark

BigDL : une règle spécifique permet l'exploitation de jupyter notebooks dédiés
(règle TCP sur port 12345)

Étape 6 : Configurer le groupe de sécurité

Un groupe de sécurité est un ensemble de règles de pare-feu qui contrôlent le trafic de votre instance. Sur cette page, vous pouvez ajouter des règles pour permettre qu'un trafic spécifique atteigne votre instance. Par exemple, si vous voulez configurer un serveur Web et permettre au trafic Internet d'atteindre votre instance, ajoutez des règles qui autorisent un accès restreint aux ports HTTP et HTTPS. Vous pouvez créer un nouveau groupe de sécurité ou en sélectionner un parmi les groupes existants ci-dessous. [En savoir plus](#) à propos des groupes de sécurité Amazon EC2.

Attribuer un groupe de sécurité: Créez un **nouveau** groupe de sécurité
 Sélectionnez un groupe de sécurité **existant**

Nom du groupe de sécurité:

Description:

Type	Protocole	Plage de ports	Source	Description
Règle TCP per	TCP	12345	Personnalisé 0.0.0.0/0	par exemple SSH for Admin Desktop
SSH	TCP	22	Personnalisé 0.0.0.0/0	par exemple SSH for Admin Desktop