TD - Outils BigData

PySpark pour la Data Science

Ouael ETTOUILEB
Machine Learning Engineer à Equancy

2024 – 2025

Université de Reims Champagne-Ardenne

Master 2 – Statistique pour l'Evaluation et la Prévision

Master 2 – Calcul Scientifique

Faisons connaissances !

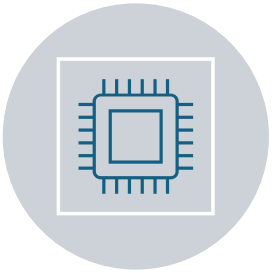
- Ouael ETTOUILEB
- Promo 2022 : M2 Statistique Pour l'Evaluation et la Prévision

- Machine Learning Engineer chez Equancy
  - Création de **modèles prédictifs** pour répondre à des **besoins business**
  - Optimisation des pipelines ML pour des solutions **fiables** et **scalables**

- Mes expériences en Data porte notamment sur :
  - Machine Learning Traditionnel (Régression, Classification et Segmentation)
  - Séries Temporelles
  - MLOps (Déploiement et intégration continue des modèles)
  - BigData (PySpark, BigQuery)
  - Outils:Python, Spark, Dataiku, Scikit-Learn, SQL, etc.

- Missions Réalisées après SEP :
  - LVMH : Sales Forecast, High Potential Detection
  - Nespresso : Promo Recommendation Engine
  - Marjane : Optimization d'Assortiment
  - La Poste Mobile : Churn Prediction

Qu’est-ce-que vous allez apprendre ?



Maîtriser **architecture et les principes de traitement** de Spark

Développer des **traitements distribués** avec Spark

Connaître les techniques de **modélisation distribuées** de Spark et construire des **pipelines de Machine Learning** et de prédiction

Adopter Spark pour réaliser vos projets data en toute **autonomie :)**

Organisation

TD1 (2H) : Introduction à Apache Spark et compréhension de ses principes.

TD2 (2H) : Développement de processus d'analyse de données et de Feature Engineering avec Spark SQL.

TD3 (2H) : Développement avancé de processus d'analyse de données et de Feature Engineering avec Spark SQL.

TD4 (2H) : Entraînement distribué d'algorithmes de Machine Learning avec Spark MLlib.

TD5 (2H) : Évaluation + Feedback sur les livrables.

Evaluation

Devoir sur table (30%)

- **QCM** sur le fonctionnement de Spark
- **Examen pratique** en Spark évaluant vos compétences en programmation et en manipulation de données

Devoirs Maison (30%)

- **À rendre chaque séance**
- Préparation d'un **notebook** après chaque séance, représentant une étape clé de traitement de données ou de Machine Learning pour votre projet final.

Projet Final (40%)

- Intégration des **remarques émises lors des code reviews** dans les notebooks rendus
- Transformation des notebooks déjà rendus en **système de Machine Learning fonctionnel**

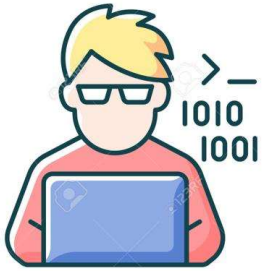
Déroulement de chaque séance

Code Review*

30 min

Théorie

30 min



Atelier d'exercices

30 min

Correction

30 min

* Pour le code review, un rendu d'un étudiant ou plusieurs seraient choisis aléatoirement pour être analysé

Projet final

- Votre mission est de développer un modèle prédictif en PySpark pour **estimer les ventes mensuelles** des magasins Walmart, en suivant les étapes suivantes :

Walmart Sales Forecast

Data Préparation — À rendre le 03/10/2024

1. Chargement des jeux de données
2. Parsing des dates
3. Nettoyage des données si nécessaire
4. Filtrage des ventes négatives
5. Agrégation des données de ventes à la maille : (Mois x produits x Magasins)
6. Ajouter les mois sans transactions et les imputer avec 0.

Basic Feature Engineering — À rendre le 17/10/2024

1. Extraction des caractéristiques temporelles
2. Attributs Produits
3. Attributs Magasins
4. Récence de chaque produit
5. Lags des ventes
6. Prix Moyen
7. Calcul de la variable cible à prédire

Advanced Feature Engineering — À rendre le 28/11/2024

1. Encodage cyclique des variables temporelles
2. Quotas de marché
3. Segmentation ABC des produits
4. Lags statistiques agrégés
5. Extraction de la de tendance
    1. Pente Reg.Lin
    2. Différences entre lags

Modélisation — À rendre le 12/12/2024

1. Split Train/Test
2. Construction d'un pipeline PySpark intégrant :
    1. One Hot Encoding
    2. Standard Scaler
    3. Un modèle de régression
    4. Optimisation des HyperParameters
3. Importance de Features
4. Calcul de métriques de performance
5. Sauvegarde du Pipeline

Livrable du projet Finale — À rendre le 21/12/2024

1. Évaluation des métriques par catégorie de produits et par région
2. Visualisation des prédictions
3. Analyse des résultats et identification des pistes d'amélioration
4. Repo GIT
5. Documentation technique (3 pages)
6. Documentation métier (1 page)

Contact

ouael@mailbox.org
https://www.linkedin.com/in/ouael/

TD 1

Introduction à Apache Spark et compréhension de ses principes

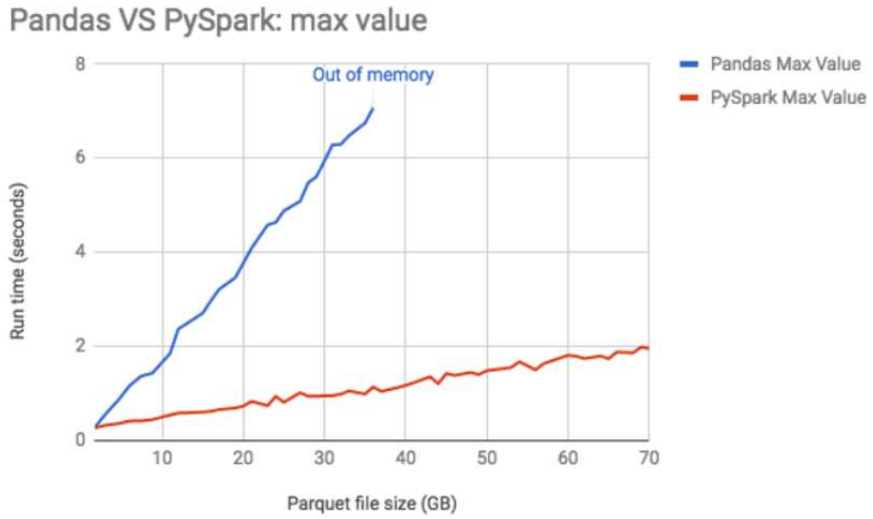
Dans cette séance

Dans TD1, vous allez apprendre :

1. Ce qu'est Spark et ses principales fonctionnalités.
2. Les différentes composantes de Spark.
3. L'architecture générale de Spark.
4. Le concept de RDD (Resilient Distributed Dataset).
5. Les principales transformations et actions sur les RDDs.

Pourquoi utiliser Spark ?

Spark permet de **traiter efficacement** de **grands volumes de données** que Pandas ne peut pas gérer, en **évitant les erreurs de mémoire insuffisante**.



- **PySpark** permet de traiter des fichiers volumineux **sans erreur**, contrairement à Pandas qui échoue avec des volumes plus petits (erreur de out of memory à partir de 36 gigas dans cet exemple).

- **PySpark maintient des performances stables et scalables**.

Apache Spark

Apache Spark est un framework de calcul distribué, rapide, tolérant aux pannes et particulièrement adapté aux environnements Big Data.

Choix Par défaut pour traitement BigData

Plus grand projet Open Source pour traitement de données

Rapide

Exécute les calculs en mémoire

Simple

Des API conviviales et de haut niveau, disponibles en Python, Scala, Java, R, SQL

Polyvalent

Supporte différents types de workloads dans un même système : batch, streaming, machine learning.

Les entreprises utilisant Spark

Analyse des données client à grande échelle

Traitement de pétaoctets de données issues des interactions des utilisateurs pour recommander des destinations personnalisées

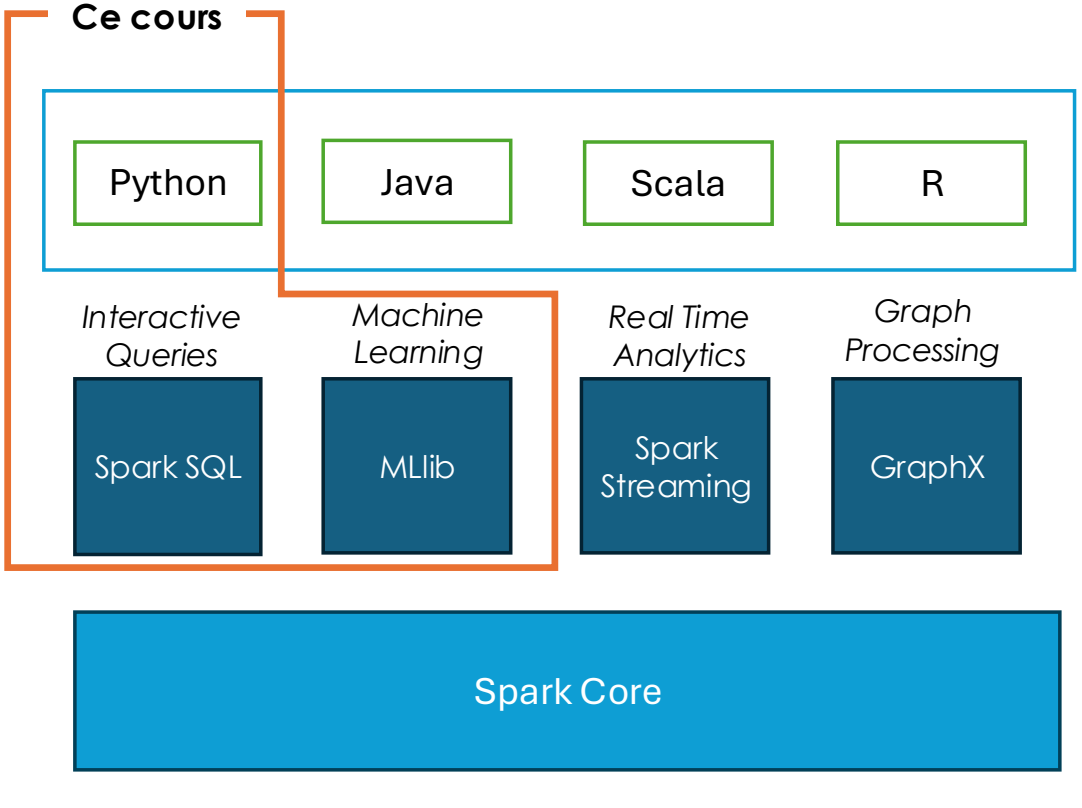
Identification des sujets et actualités les plus populaires grâce au machine learning

traitement de flux en temps réel afin de fournir des recommandations personnalisées à ses clients.

Identification des tendances des utilisateurs

Recommandation de recettes et d'aliments

Les composants de l'écosystème Spark



- **Les APIs de Spark sont multi-langages**
- Spark offre des API multi-langages (Spark SQL, Streaming, etc.) en Python, Java, Scala et R.

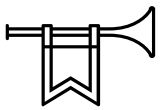
- **Apis de haut niveau**
- Facilitant l'analyse de données en fournissant des abstractions puissantes, optimisées et simple à utiliser.
- Elles sont construites sur Spark Core et s'appuie sur les RDD

- **API de bas niveau**
- Fournit les éléments fondamentaux pour le traitement distribué des données :
  - RDD *(Structure de Données de bas niveau distribuées tolérante aux pannes)*
  - Gestion de l'infrastructure de calcul distribué (Coordination, Scheduling, et distribution sur plusieurs machines)

Programmation Spark

Deux options s'offrent pour développer une application Spark :

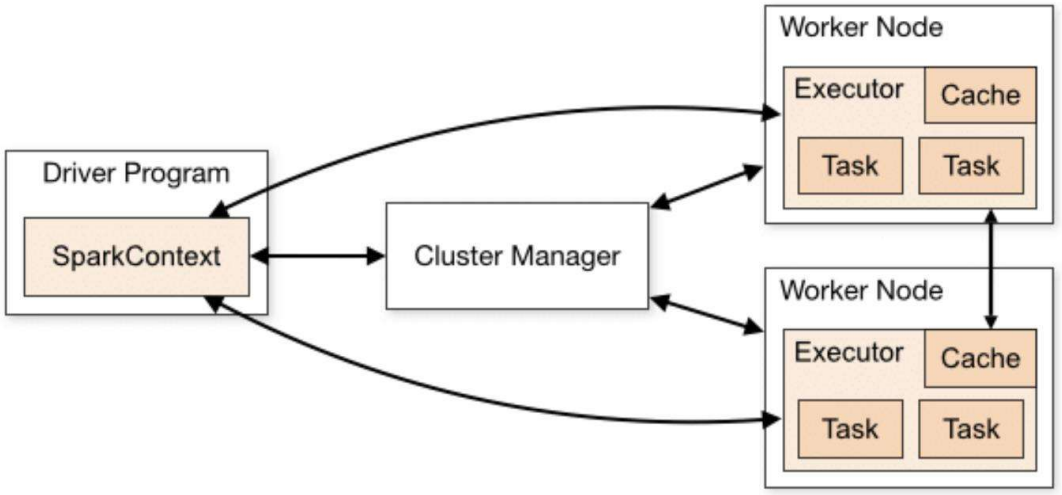
- **Programmation bas niveau :** Utilisation d'opérations sur une structure de données de bas niveau appelée Resilient Distributed Dataset (RDD).
- **Programmation haut niveau :** Utilisation de bibliothèques haut niveau, telles que Spark SQL et Spark Streaming.



Dans TD1, nous allons se concentrer sur la programmation bas niveau pour mieux comprendre le fonctionnement interne de Spark.

À partir de TD2 nous allons nous concentrer uniquement sur la programmation haut niveau en utilisant les APIs Spark SQL et Spark MLlib ☺

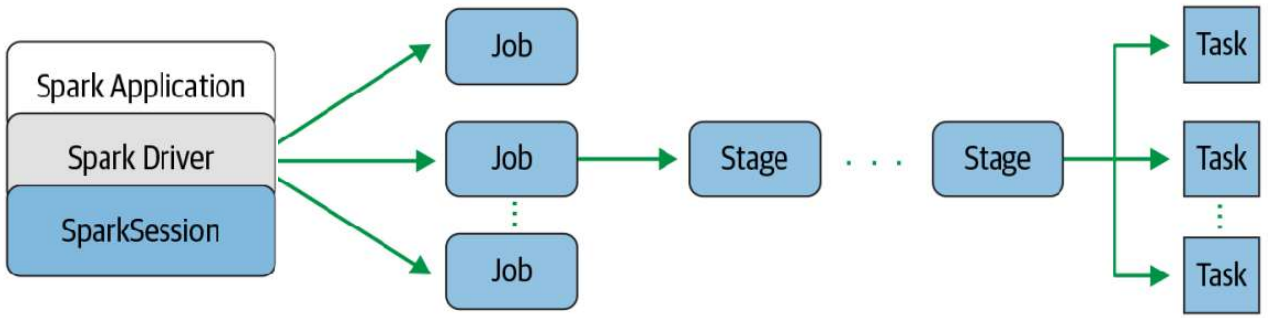
Architecture générale de Spark



Composant	Définition
Driver Program	Le driver un processus tourne sur un Master Node et qui est responsable **de transformer les opérations en un DAG** (Directed Acyclic Graph) de calculs. Il **planifie** et **distribue** les tâches sur les worker nodes, **coordonne** l'exécution, **et récupère les résultats**
Cluster Manager	Processus responsable de l'allocation des ressources nécessaires (Worker, CPU, RAM) pour exécuter une application Spark. Spark supporte quatre types de gestionnaires de clusters : **standalone**, **YARN**, **Mesos** et **Kubernetes**.
Executor	Un executor est un processus lancé sur un worker node qui **exécute les tâches attribuées par le driver**
Worker Node	Un worker node est une **machine physique ou virtuelle** qui héberge les processus executor pour exécuter les tâches et stocker les données intermédiaires.

Anatomie d'une application Spark

Composant	Définition
Spark Application	est un **programme écrit en Spark** (via des API comme PySpark – Spark SQL / MLlib) qui s'exécute sur un cluster Spark. Elle inclut un **Driver** qui gère la logique de l'application et un ou plusieurs **Executors** qui effectuent les tâches de calcul.
Spark Session	**Point d'entrée** principal pour interagir avec Spark et accéder aux APIs permettant d'utiliser ses fonctionnalités.
Job	Un Job est une unité de travail **déclenchée par une action** (comme collect(), save(), count()) dans une application Spark. Chaque action dans un Spark DataFrame ou un RDD lance un ou plusieurs Jobs.
Stage	**Spark divise un job en plusieurs stages**, chacun étant constitué d'une série **de transformations qui n'impliquent pas de shuffle** (c'est-à-dire pas de redistribution de données entre les nœuds).
Task	Une Task est la plus petite unité d'exécution dans Spark. Un **stage est divisé en plusieurs tâches**, et chaque tâche est **exécutée sur une partition d'un RDD ou DataFrame par un Executor**.



Resilient Distributed Dataset (RDD)

Un Resilient Distributed Dataset (RDD) est une collection **distribuée immuables** d'éléments de données, répartis sur les nœuds d'un cluster Spark.

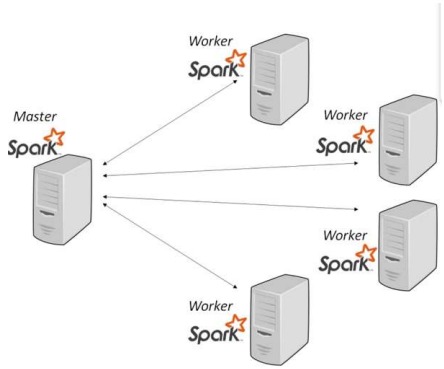
Resilient

Tolérance aux pannes

Grâce à un mécanisme d'auto-récupération des données, en utilisant un **graphe acyclique dirigé (DAG)** qui permet de reconstruire les données en cas de défaillance d'un nœud.

Distributed

Le calcul des RDD est distribué sur plusieurs nœuds



Dataset

Collection de données partitionnée

Liste

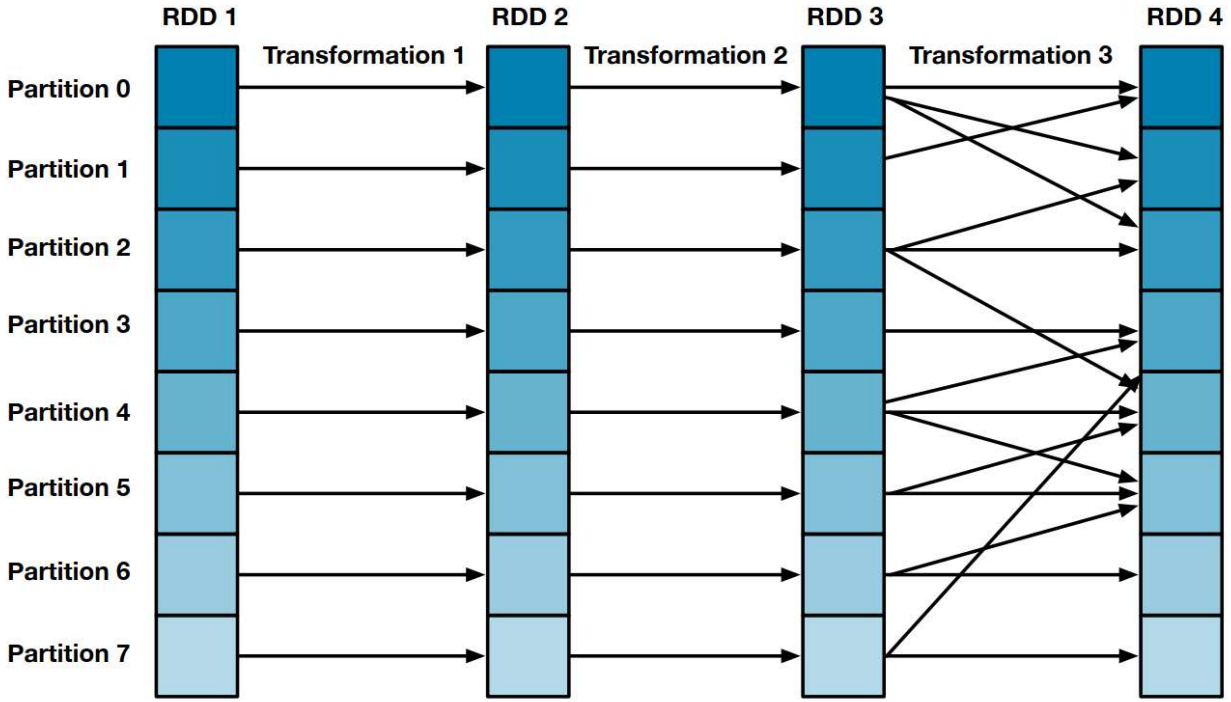
A
B
C
D
E
F

RDD

A	Partition 0
B	
C	Partition 1
D	
E	Partition 2
F	

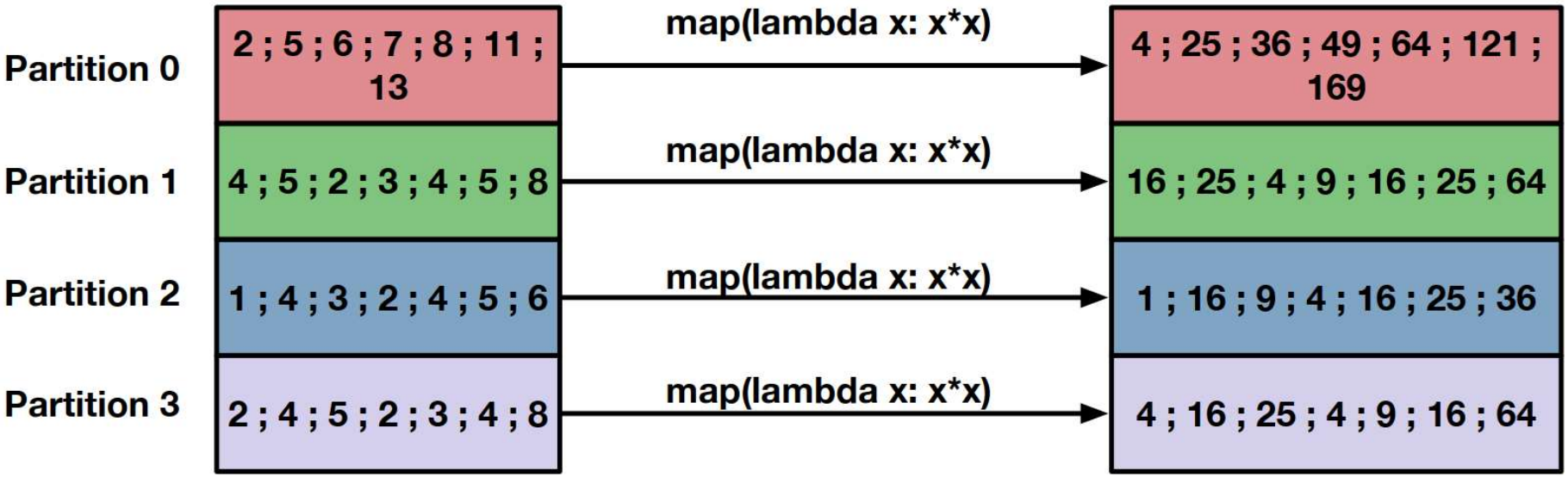
RDD Transformation

- Une transformation est une opération qui prend en entrée un ou plusieurs RDDs et retourne un nouveau RDD.
- Une transformation est **appliquée en parallèle sur chaque partition**.
- **Les transformations sont paresseuses** ne renvoient aucun résultat concret, elles s'accumulent. Elles sont exécutées seulement lorsque on appelle une action (collect(), count()..)



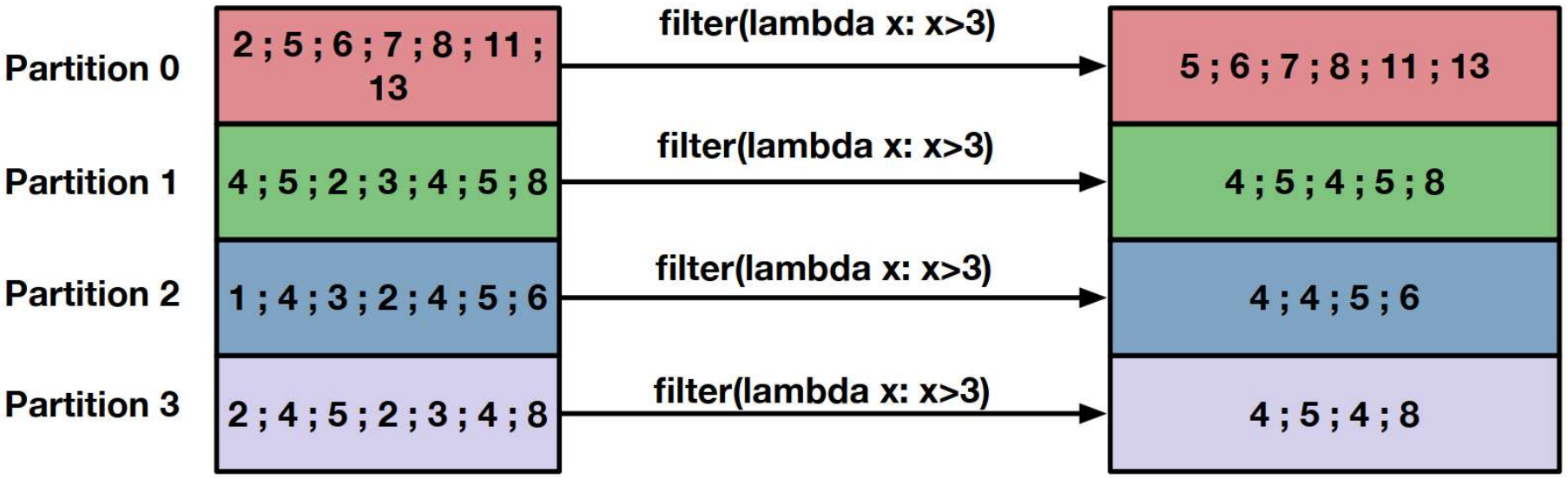
RDD Transformation: map

Renvoie un nouveau RDD en appliquant une fonction à chaque élément de cet RDD



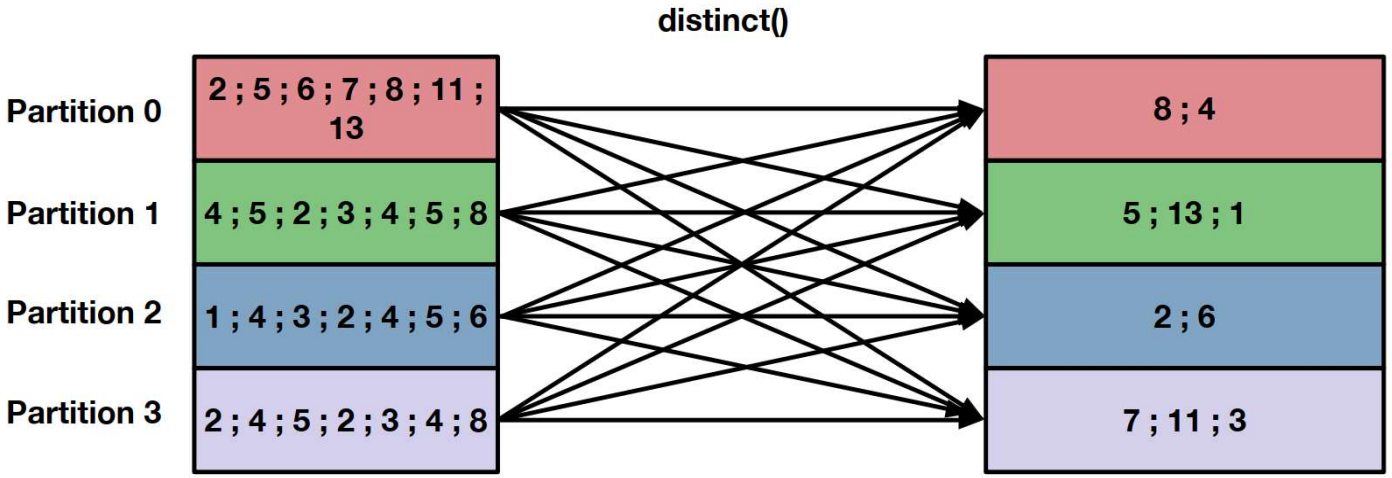
RDD Transformation : Filter

Renvoie un nouveau RDD contenant uniquement les éléments qui satisfont une condition



RDD Transformation : Distinct

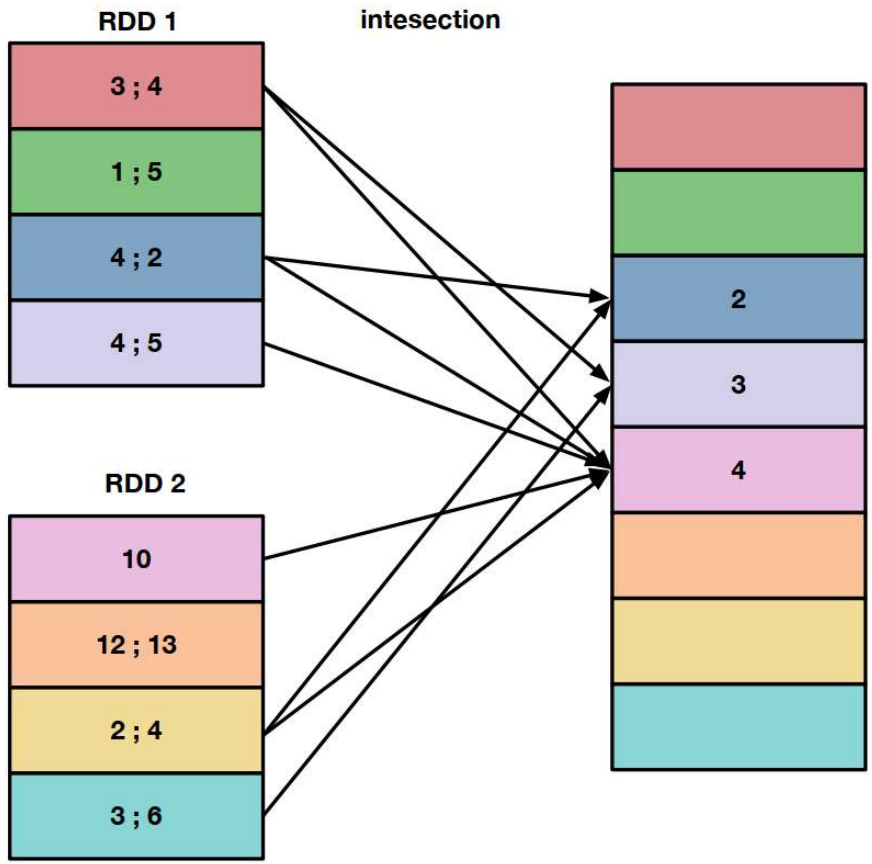
Renvoie un nouveau RDD contenant les éléments distincts de ce RDD.



Contrairement aux transformations précédentes, distinct entraîne un réarrangement (shuffle) des données à travers les partitions

RDD Transformation : intersection

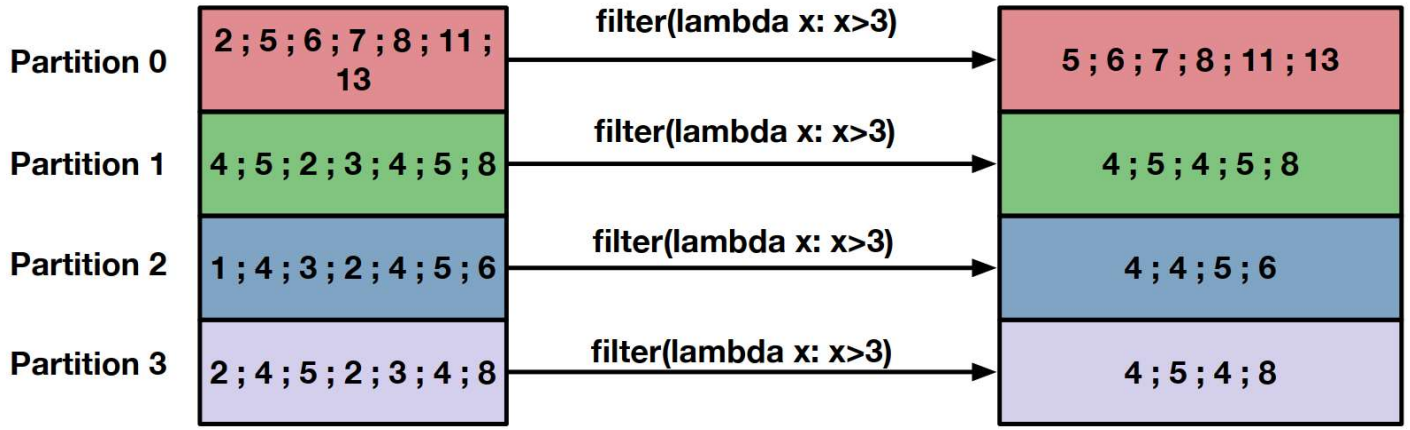
Intersection prend en entrée un ou deux RDDs et renvoie un nouveau RDD contenant les éléments présents dans les deux RDDs.



Narrow Transformation

- Narrow transformation est une transformation **où chaque partition du RDD de sortie dépend seulement d'une partition du RDD d'entrée**.

- Les opérations telles que **filter**, **map**, **flatMap**, **et union** sont des transformations Narrow.

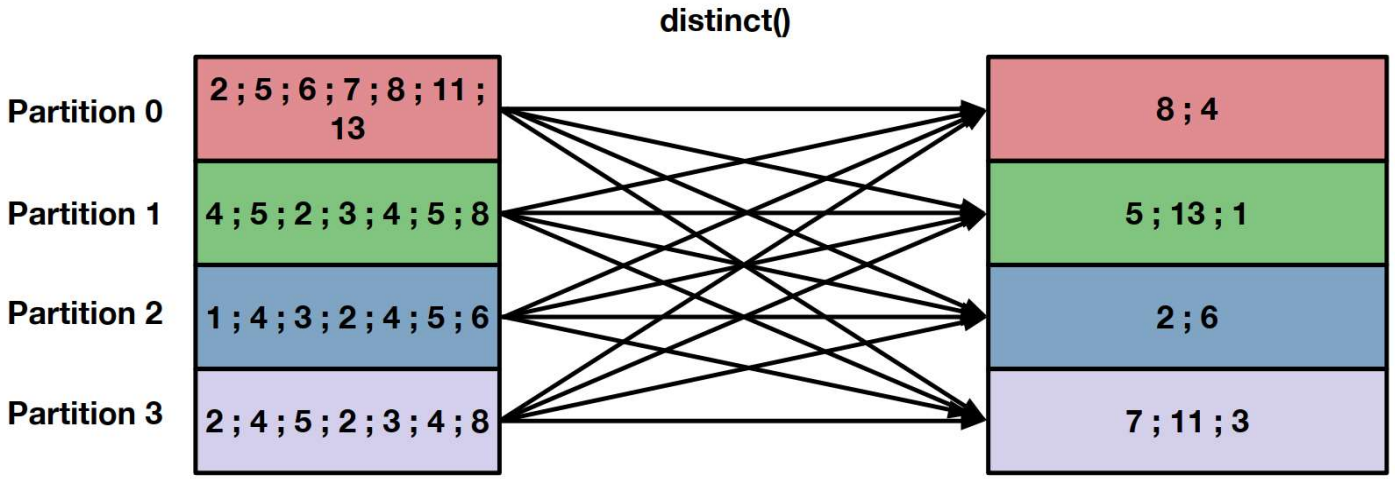


- Ces transformations sont **peu coûteuses** car elles **ne nécessitent pas de communication entre les exécuteurs** (executors).

Wide Transformation

- Une **transformation Wide** est une opération **où chaque partition du RDD de sortie peut dépendre de plusieurs partitions du RDD d'entrée**.

- Les transformations comme **distinct et intersection** sont des exemples de transformations Wide.



- **Elles sont plus coûteuses** : Elles demandent plus de ressources car les exécuteurs doivent communiquer entre eux.

- **Provoquent un réarrangement des données** : Ces transformations entraînent un transfert (shuffle) des données à travers le réseau du cluster.

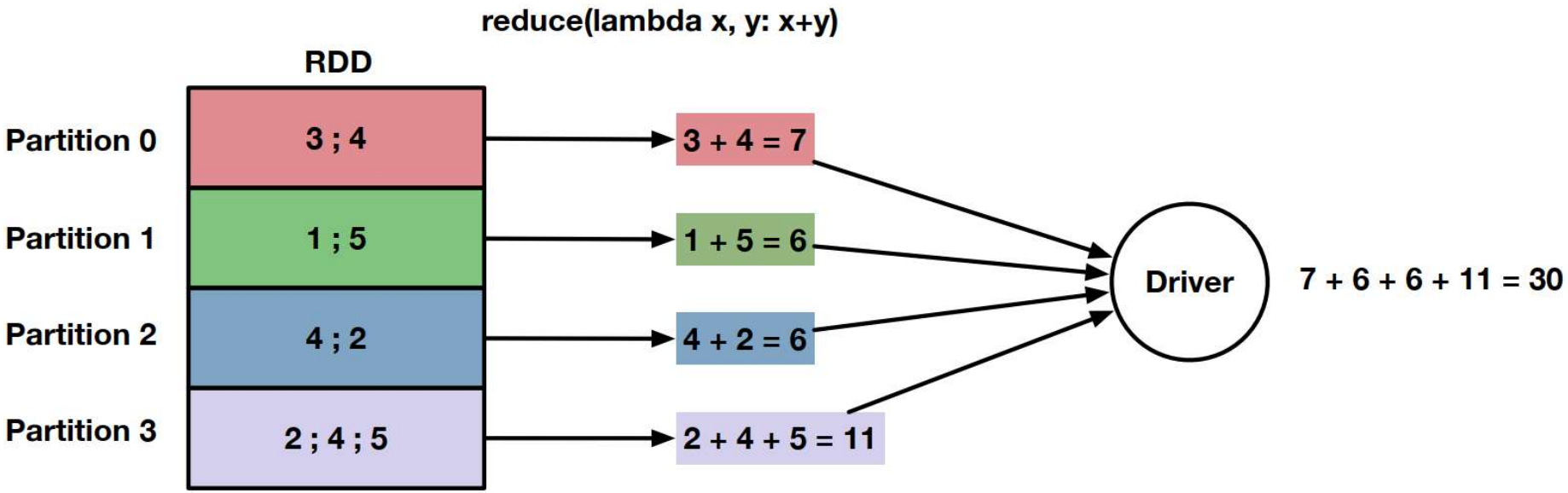
RDD Actions

- Une action est une opération qui déclenche l'exécution du DAG (Directed Acyclic Graph) des transformations pour produire un résultat ou enregistrer des données sur un stockage persistant.

- Contrairement aux transformations, **les actions retournent des résultats** concrets, soit au driver, soit sur un système de stockage externe.

RDD Actions : reduce

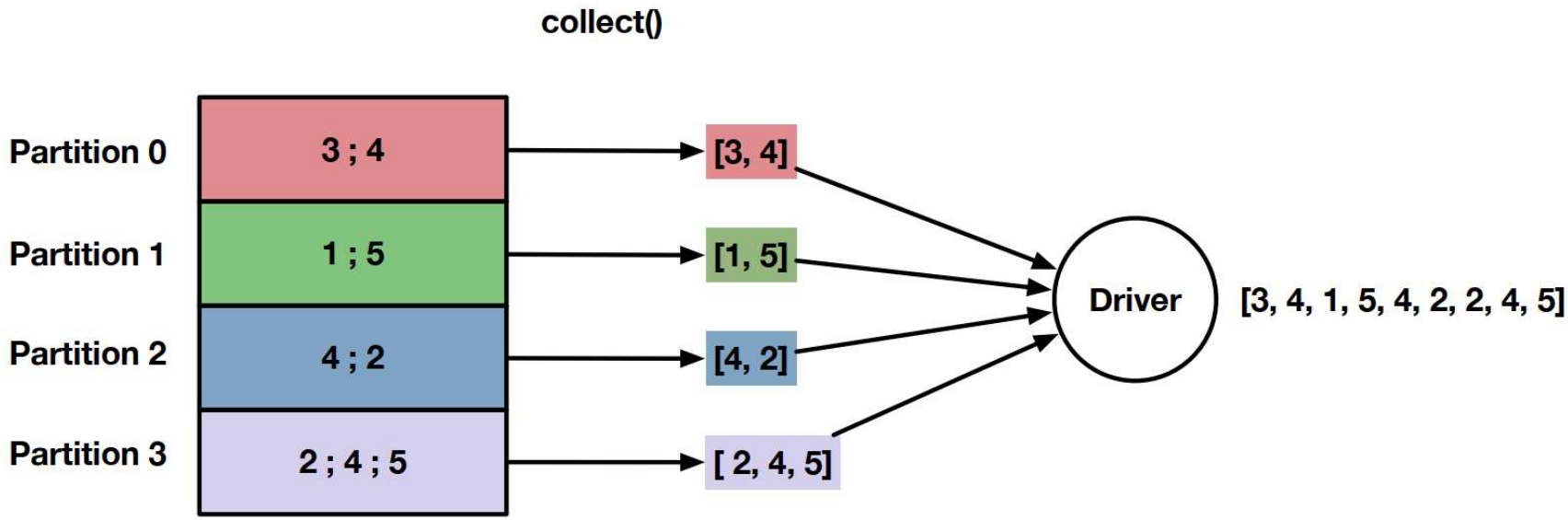
reduce réduit l'ensemble des éléments du RDD à une seule valeur en appliquant une fonction successivement à deux éléments à la fois.



RDD Actions : collect

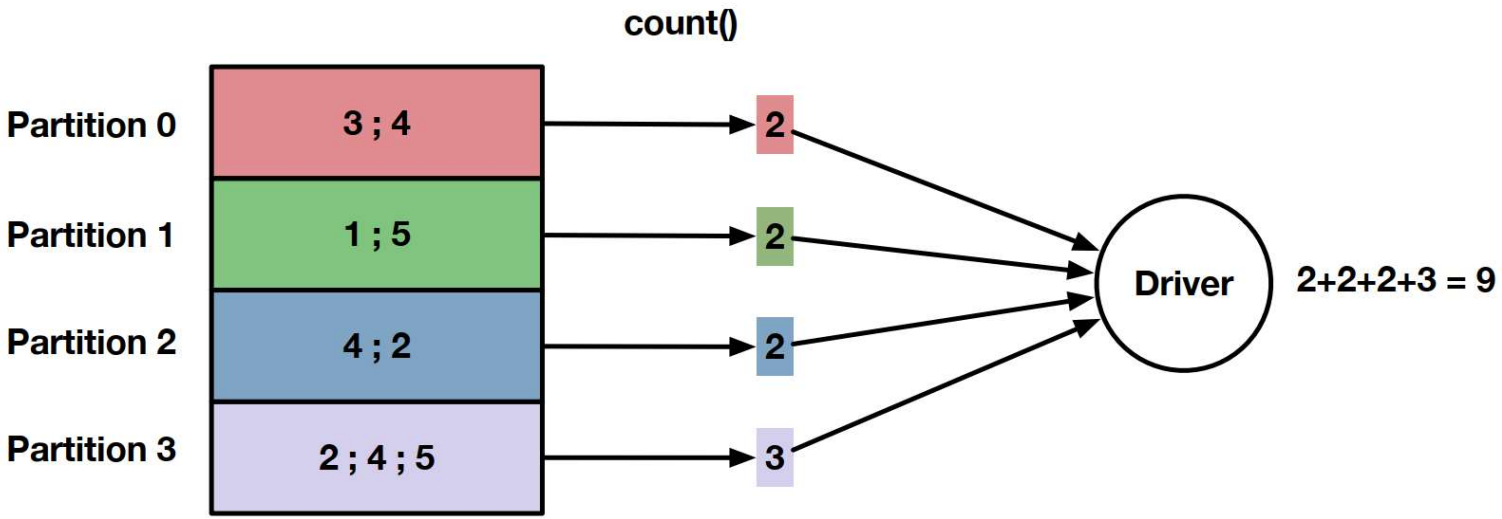
- Lorsqu'on exécute collect(), Spark exécute toutes les transformations sur les données, puis ramène tous les éléments du RDD au driver.

- Cette action est utile lorsqu'on souhaite accéder à tous les résultats d'un calcul distribué sur la machine locale pour les analyser ou les utiliser dans un autre traitement.



RDD Actions : count

Renvoie le nombre d'éléments dans un RDD



Liste des transformations à connaitre

Transformation	Description
map(func)	Renvoie un nouveau RDD en appliquant une fonction à chaque élément de cet RDD
flatMap(func)	Comme map, mais peut retourner plusieurs éléments pour un seul élément d'entrée, ce qui permet d'aplatir les résultats.
filter(func)	Retient uniquement les éléments qui satisfont la condition de la fonction func.
reduceByKey(func)	Combine les valeurs ayant la même clé en utilisant la fonction associative et commutative func.
groupByKey()	Regroupe toutes les valeurs ayant la même clé.
sortByKey()	Trie les paires clé-valeur par clé.
join(otherRDD)	Réalise une jointure entre deux RDD sur les clés communes..
union(otherRDD)	Combine deux RDD en un seul, contenant tous les éléments des deux RDD.
intersection(otherRDD)	Renvoie un RDD contenant les éléments communs aux deux RDD.
distinct()	Supprime les doublons dans le RDD..
sample(withReplacement, fraction)	Échantillonne une fraction des données du RDD, avec ou sans remise.
mapValues(func)	Applique une fonction aux valeurs de chaque paire clé-valeur sans modifier les clés.
coalesce(numPartitions)	Réduit le nombre de partitions
repartition(numPartitions)	Modifie le nombre de partitions en provoquant un shuffle complet des données.
cartesian(otherRDD)	Renvoie le produit cartésien de deux RDD. Peut être très coûteux en termes de performance et de mémoire.

Liste des actions à connaitre

Action	Description
collect()	Récupère tous les éléments du RDD sur le driver. À utiliser avec prudence pour les grands RDD, car peut entraîner une saturation de la mémoire du driver.
count()	Renvoie le nombre total d'éléments dans le RDD.
take(n)	Récupère les n premiers éléments du RDD. Utile pour prévisualiser les données.
reduce(func)	Agrège les éléments du RDD en utilisant une fonction func associative et commutative.
first()	Renvoie le premier élément du RDD.
countByKey()	Pour un RDD de paires clé-valeur, compte le nombre d'éléments pour chaque clé.
saveAsTextFile(path)	Sauvegarde le RDD sous forme de fichiers texte dans le répertoire spécifié. Chaque partition est sauvegardée comme un fichier distinct.
saveAsSequenceFile(path)	Sauvegarde le RDD sous forme de fichiers séquence Hadoop. Utile pour l'interopérabilité avec d'autres outils Hadoop.
foreach(func)	Applique la fonction func à chaque élément du RDD. Les opérations sont exécutées sur les nœuds de travail, pas sur le driver.

S’amuser avec les transformations et les actions sur Google Colab