Formation Spark

Ihab ABADI / UTOPIOS

Programme Jour 1

- Introduction Spark
- Les composants de Spark
- Api RDD
- Les transformations
- Les actions
- Fonctionnement interne
- RDD Key Value
- Utilisation des filtres et flatmap
- Principe de SparkSQL
- Utilisation de SparkSQL
- Utilisation des DataSets

Programme Jour 2

- Les fonctions udf
- Utilisation des Broadcast variables.
- Utilisation des Accumulators.
- Persistance dans Spark.
- Introduction à Spark Streaming
- Utilisation de Dstream API
- Utilisation des Structured Streaming

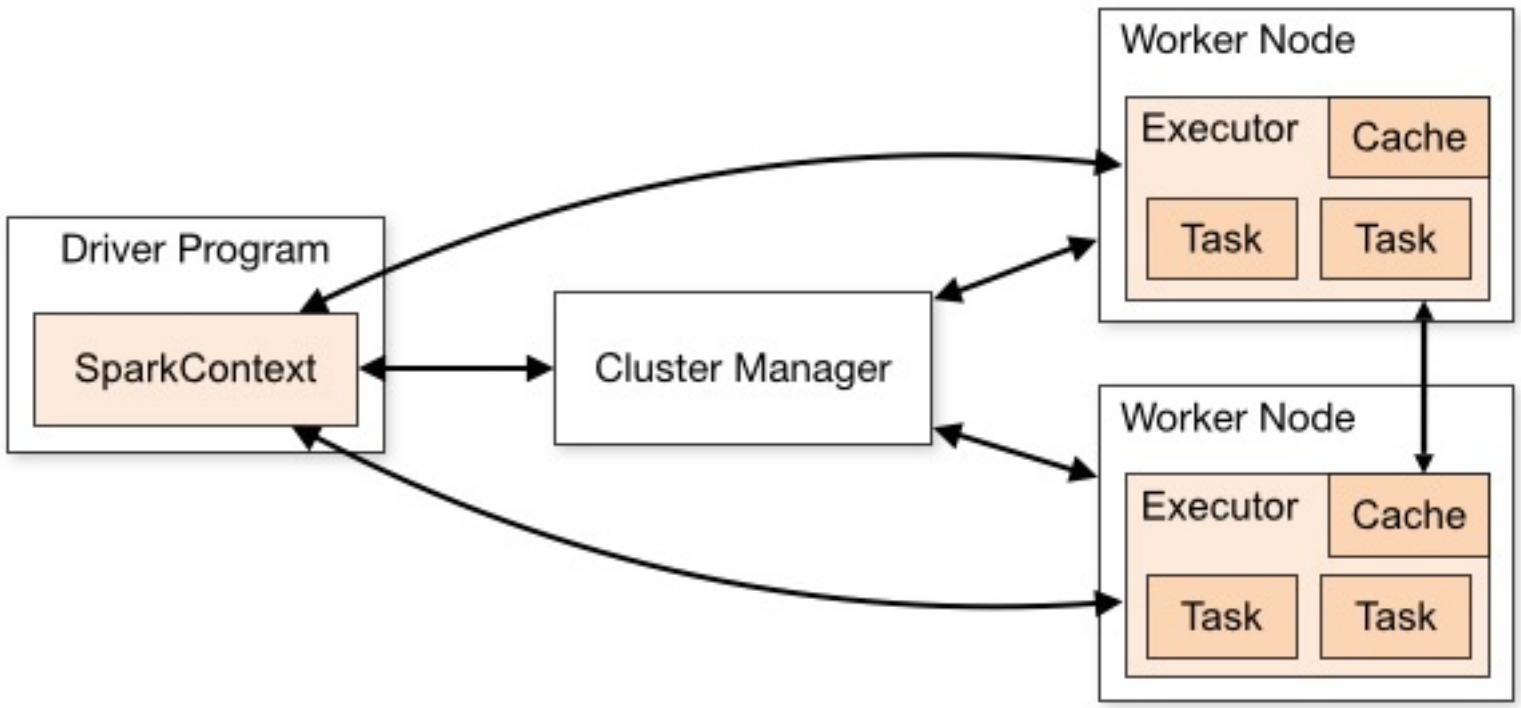
Programme Jour 3

- Utilisation de spark submit.
- Utilisation des API Rest de Spark
- Le partitionning
- Bonnes pratiques sur un cluster Spark
- Introduction à la MLLib
- La régression linéaire avec MLLib
- Etude de cas

Introduction Spark

- Spark est un moteur de traitement de données à grande échelle.
- Spark permet d'analyser et transformer une grande quantité de données.
- Spark permet l'exécution et l'analyse sur un cluster.
- Spark est 100 fois plus rapide que Hadoop MapReduce
- Spark utilise DAG Engine pour l'optimisation des workflows.

Introduction Spark



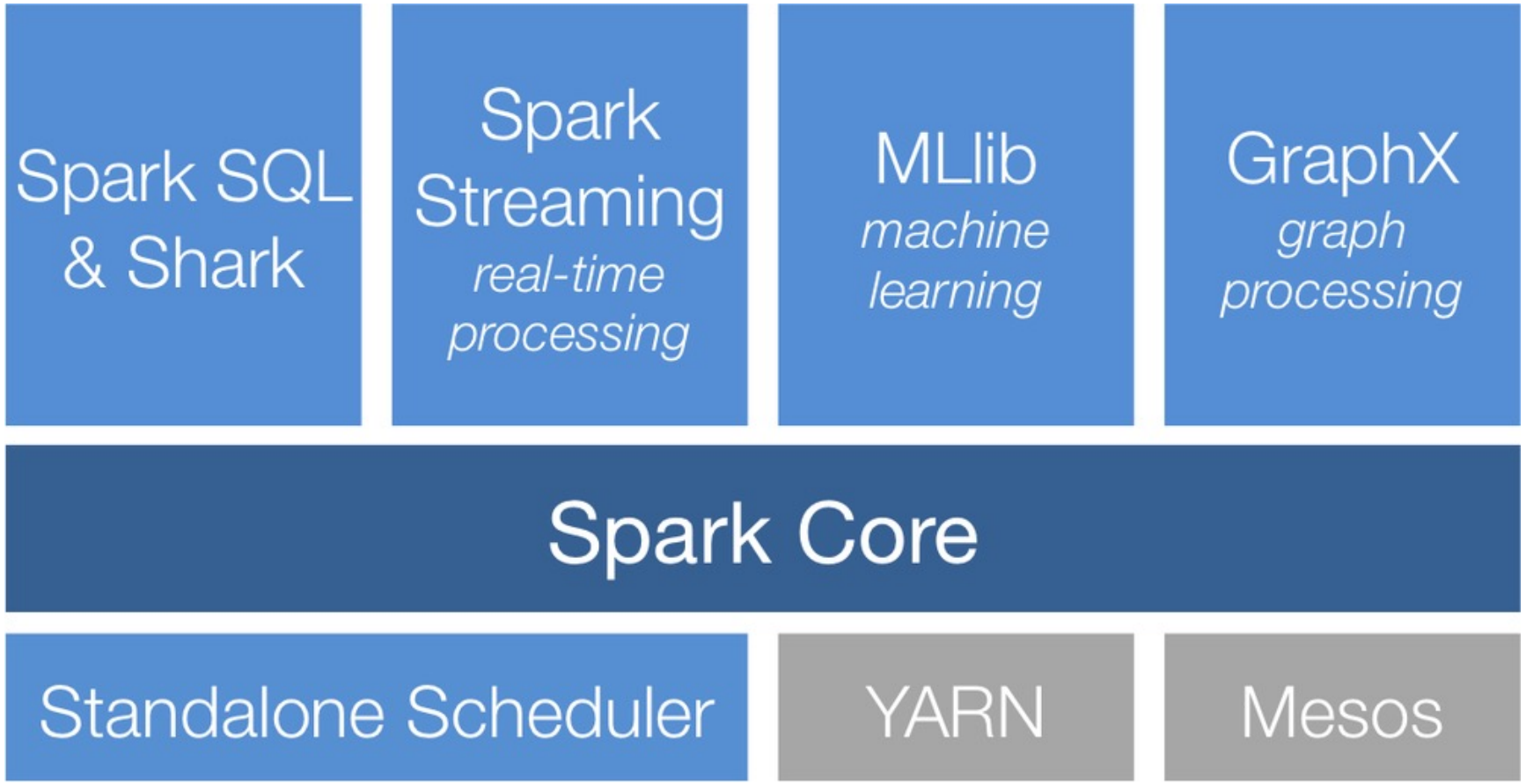
Introduction Spark

- Driver program est un script ou une application qui peut être en Java, scala, python.
- Cluster manager est l'api qui permet de gérer l'orchestration de notre cluster de nœuds.
- Chaque nœud contient un exécuteur.
- Chaque exécuteur possède son propre cache et la liste des tâches à exécuter.
- Les composants de l'architecture spark communiquent et se synchronisent entre eux.

Introduction Spark

- Spark est facile à apprendre.
- Spark supporte une multitude de langages de développement.
- Spark fournit des APIs bas niveau.
- Spark fournit également des Apis Haut niveau.

Les composants de Spark



Les composants de Spark

- Spark Core est le moteur d'exécution de Spark.
- Spark Streaming est une Api qui permet l'ingestion de données en temps réel.
- Spark SQL permet l'interaction avec Spark à l'aide de requête SQL.
- MLLib est la bibliothèque d'apprentissage automatique de Spark.
- GraphX est une Api qui permet de mettre en place et analyse de connexion entre plusieurs entités.

Nouveauté spark 3

- Dépréciation MLLib basé sur les RDD.
- Le partition dynamique pour améliorer les performances de Spark (17X).
- Dépréciation de Python 2.
- Utilisation des GPU.
- Intégration de Kubernetes pour l'orchestration.
- Support des fichiers binaires.
- Utilisation de Cypher dans SparkGraph qui remplacera GraphX.

Tools à utiliser

- Spark 3
- Java > version 8
- IntelliJ ultimate.

Spark Core – RDD (Resilient Distributed DataSet)

- RDD sont des lignes de données muables qui sont :
  - Resilient
  - Distributed
  - DataSet
- Les RDD sont créées par notre Driver Program.
- La création se fait à l'aide d'un context (SparkContext).

Spark Core – RDD

```
JavaSparkContext sc = new JavaSparkContext( master: "local[*]",  appName: "app");
JavaRDD<Integer> rdd = sc.parallelize(Arrays.asList(1,2,3,5,7,8));
```

- La création peut se faire à partir d'un fichier text.
  - File://, s3n://, hdfs://
- La création peut se faire également à partir d'une base de données
  - JDBC
  - Cassandra
  - ElasticSearch
  - JSON, csv….

Spark core- Transformations

- Les opérations appliquées aux RDD sont des transformations.
- Chaque transformation donne une nouvelle RDD.
- Spark core offre une liste de transformations possibles :
  - Map
  - Flatmap
  - Filter
  - Distrinct
  - Sample
  - …
- Chaque transformation prend comme paramètre une fonction (Expression lambda)
- Une démo.

Spark core- Transformations

- Spark Context ne procède pas directement à l'exécution de la transformation.
- Spark attend l'appel au deuxième type d'opération exécutable sur RDD (action) pour démarrer la transformation.
- Le choix de l'action influence sur la façon d'exécution de la transformation.

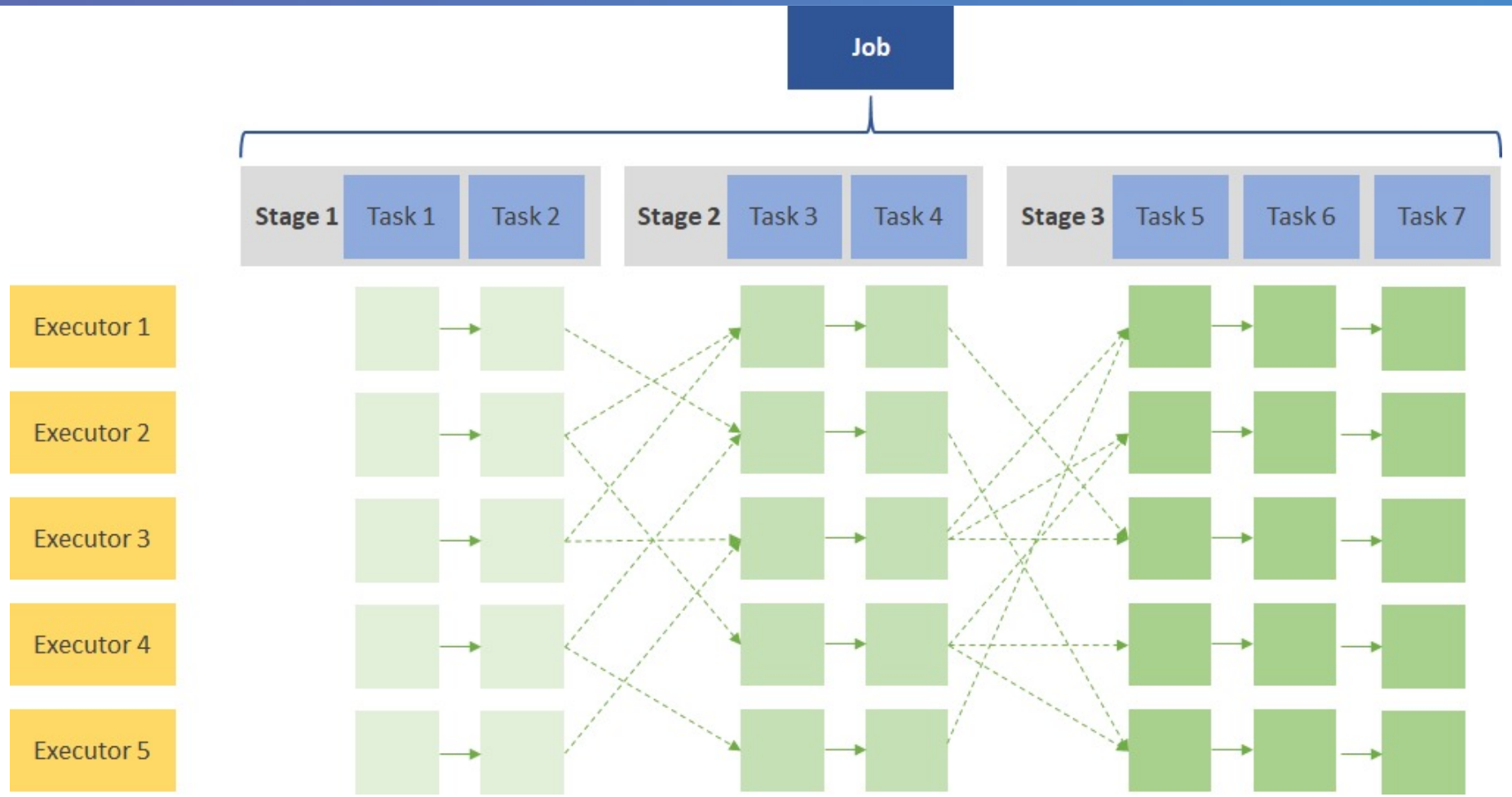
Spark core- Actions

- L'action est l'opération à appliquer à notre RDD pour récupérer le résultat.
- Spark core offre une liste d'actions possibles.
  - Collect
  - Count
  - CountByValue
  - Take
  - Top
  - Reduce
  - ….
- Une action peut avoir aucun ou plusieurs paramètres.
- Démo

Spark core – Rdd – Exemple 1

- On utilisera un set de données de plus de 100 000 lignes de notes de films (fichier film.data).
- Dans ce fichier chaque film est noté de 1 à 5.
- On souhaite connaitre le nombre de film par note et faire un tri.
- Le fichier film.data contient sur chaque ligne l'identifiant de l'utilisateur, l'identifiant du film, la note et le timetamp de validation.
- On commence par extraire à partir de chaque ligne la valeur qui nous intéresse.
- On compte le nombre d'éléments par valeur.
- On tri le résultat.
- Démo.

Spark core – fonctionnement interne



Spark core – RDD Key Value

- RDD key value est une implémentation des données structurées sous format clé valeur.
- RDD key value offre des transformations et des actions en plus.
- Exemple:
  - Pour calculer le nombre moyen d'amis par âge dans un set de données qui contient (personnes avec nombre d'ami).
  - On commence par extraire les données dans une RDD (âge, ami).
  - On applique les transformations mapValues, reducesByKey
  - On applique une action de collect
- Démo

Spark core – RDD Transformation filter

- Transformation Filter est une opération qui prend comme paramètre une fonction.
- La fonction paramètre est appliquée à chaque élément et renvoie un boolean.
- Le résultat est une nouvelle RDD avec des éléments dont le résultat est positif.
- Exemple:
  - A partir d'un set de données météorologique fournit par deux stations météo, on souhaite conserver uniquement les températures minimums.

Spark core – RDD flatmap

- La fonction map permet de convertir chaque ligne de notre set de données en ligne de RDD.
- Avec map chaque ligne est une row RDD.
- FlatMap permet de convertir chaque ligne de notre set de données en n'importe quel nombre de row RDD.

- Exemple :
  - A partir d'un fichier texte, on souhaite compter le nombre de mots.

Exercice 1

- A partir d'un fichier csv qui contient sur chaque ligne (id client, id article, montant).
- Ecrire un driver programme qui permet de connaître le montant dépensé par chaque client.

Exercice 1 - aide

- Séparer chaque ligne avec la délimitation « , ».
- Créer une pair RDD avec id client et montant (transformation map).
- Réduire par id client (transformation reduceByKey).
- Récupérer une liste de résultats (action collect).

SPARK SQL- RDD Structurés

- Un RDD est une ”boîte” (typée) destinée à contenir n’importe quel document, sans aucun préjugé sur sa structure (ou son absence de structure)
- Cela rend le système très généraliste, mais empêche une manipulation fine des constituants des documents, comme par exemple le filtrage en fonction de la valeur d’un champ.
- C’est au programmeur de fournir la fonction effectuant le filtre.
  Spark propose (depuis la version 1.3, avec des améliorations en 1.6 puis 2.0) des RDD structurées dans lesquels les données sont sous forme tabulaire. Le schéma de ces données est connu.
- Ces structures, Datasets et Dataframes, sont assimilables à des tables relationnelles.

SPARK SQL - DataSet et DataFrames

- La connaissance du schéma - et éventuellement de leur type - permet à Spark de proposer des opérations plus fines, et des optimisations inspirées des techniques d'évaluation de requêtes dans les systèmes relationnels.
- On se ramène à une implantation distribuée du langage SQL.
- En interne, un avantage important de la connaissance du schéma est d'éviter de recourir à la sérialisation des objets Java (opération effectuée dans le cas des RDD pour écrire sur disque et échanger des données en réseau).
- Ces RDDs structurés sont nommés :
  - Dataset quand le type des colonnes est connu
  - Dataframe quand ce n'est pas le cas
- Un Dataframe n'est rien d'autre qu'un Dataset contenant des lignes de type Row dont le schéma précis n'est pas connu

SPARK SQL – DataSet/DataFrames Exemple

- Pour utiliser SparkSQL, on peut créer un point d'entrée de type SqlContext.
- On peut également, depuis la version 2, utiliser le point d'entrée SparkSession.
- Exemple:
  - En utilisant notre set de données friends, on souhaite extraire les personnes âgées de 20 à 25 ans.
- Etapes:
  - Création du context
  - Extraction des lignes dans une RDD
  - Création d'une View.
  - Exécution d'une requête SQL sur la view.
  - Récupération des résultats.

SPARK SQL – DataSet/DataFrames Exercice 1

- On souhaite reprendre notre exemple moyenne d'amis en fonction de l'âge « Section RDD (Key/Val) » avec une utilisation des DataSets.
- Rappel :
  - Chaque ligne du fichier est composée de :
    - Identifiant de l'utilisateur.
    - Nom de l'utilisateur.
    - Age de l'utilisateur.
    - Nombre d'ami.

SPARK SQL – DataSet/DataFrames Exercice 1- Aide

- Création d'une sparkSession.
- La lecture du set de données, en utilisant la ligne d'entête.
- La sélection des colonnes.
- L'utilisation des fonctions avg, groupby et show.

SPARK SQL – DataSet/DataFrames Exercice 2

- Reprendre L'exercice 1 (Montant dépensé par client) avec les dataSets.