Université Mohammed V- Rabat
Ecole Mohammadia d'Ingénieurs
Département Génie Informatique
Filière Génie Informatique et Digitalisation

Traitement Big Data

Pr. N. EL FADDOULI
nfaddouli@gmail.com

2023-2024

Chapitre 1: Présentation d'Apache Spark

- Historique du Framework.
- Les versions de l'API Spark (Scala, Python et Java).
- Les différents modules de Spark.
- Comparaison avec l'environnement Apache Hadoop.

Présentation d'Apache Spark: Historique

- Le **framework Spark** a démarré comme un projet de recherche au AMPLab (ancien R&D Lab) à l'université de Californie Berkley en 2009.
- Spark est un framework pour le **calcul distribué** appliqué en Big Data.
- Il est devenu open source en 2010.
- Il a été transféré à Apache Software Foundation en 2013 pour devenir, en 2014, un projet Apache de haut niveau.
- En 2013 la société **Databricks** est fondée par les créateurs de Spark afin d'offrir une plate-forme Web pour l'utiliser.
- Spark a été conçu avec un important changement d'esprit par rapport à Hadoop où les résultats de calcul sont sur disque: **les résultats des intermédiaires de Spark sont dans la RAM.**

Présentation d'Apache Spark: Composants (1/5)

- Spark est écrit en **Scala** (*langage fonctionnel*).
- Spark fournit des **API** de haut niveau pour les langages de programmation **Java**, **Scala**, **Python** et **R**
- Scala est le langage natif de Spark: toute nouvelle API est disponible en premier en Scala.
- Spark prend en charge **SQL**, le **streaming** de données, l'apprentissage automatique (**ML**) et le traitement de **graphes**.

Présentation d'Apache Spark: Composants (2/5)

❑ **Moteur d'exécution** général pour la plate-forme SPARK sur **lequel toutes les autres fonctionnalités sont construites**.

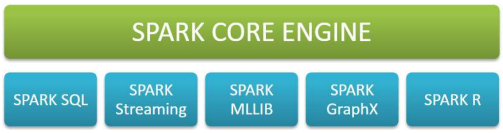
• Il a la responsabilité de **planifier** et **répartir** les **tâches** sur le **cluster**, de **coordonner** les opérations d'entrée/sortie ou encore de **récupérer** les éventuelles **pannes**.

• Il fournit le **traitement en mémoire**.

❑ Spark comprend un ensemble d'outils et de **modules de haut niveau** tels que :

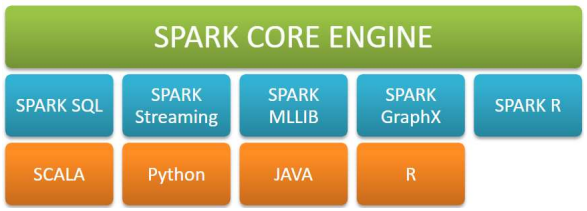
- **Spark SQL** est destiné au traitement de données structurées à l'aide de SQL et à l'accès à des sources de données externes comme les **SGBDR** ou **Hive** (avec HQL).
- **Spark STREAMING** permet d'effectuer des analyses de données en streaming (***flux de données à la volée***). Il prend en charge des données venant de différentes sources telles que Flume, Kinesis ou Kafka.
- **Spark MLib** : c'est la bibliothèque intégrée d'Apache Spark pour le ML. Il fournit plusieurs algorithmes d'apprentissage automatique ainsi que plusieurs outils permettant de créer des pipelines de ML.

Présentation d'Apache Spark: Composants (3/5)



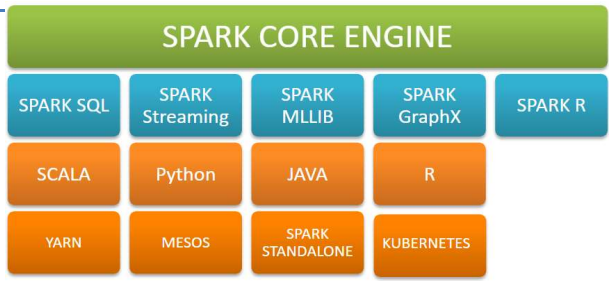
- **Spark GraphX** permet d'effectuer des modélisations, des calculs et des **analyses de graphe** au sein d'une architecture distribuée (calcul parallèle).
- **Spark R** est un package qui permet d'utiliser Spark à partir du langage **R**.

❑ Spark fournit des **API de haut niveau** pour les langages de programmation **Java**, **Scala**, **Python** et **R**



Présentation d'Apache Spark: Composants (4/5)

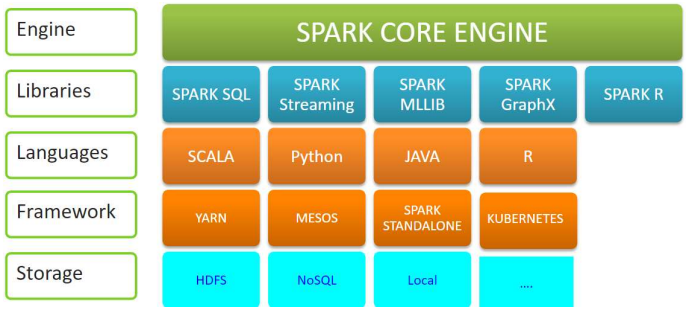
❑ Spark peut s'appuyer sur plusieurs **gestionnaires de cluster** afin d'avoir les **ressources** nécessaires pour exécuter ses tâches.



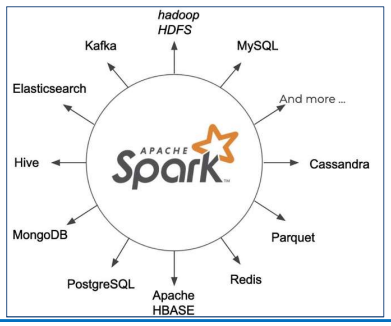
- Il peut utiliser son propre gestionnaire de cluster intégré qui permet de déployer (en mode **Standalone**)) et de gérer des clusters Spark pour le traitement distribué des données.
- Le mode **Standalone** est relativement facile à mettre en place, ce qui le rend approprié pour les environnements de développement, de test et de production.
- Bien qu'il puisse gérer des clusters de taille variable, Spark **Standalone** est souvent utilisé pour des configurations de clusters plus petites ou pour le développement et les tests.
- On peut également déployer et utiliser Spark en **local** pour l'apprentissage et le développement.

Présentation d'Apache Spark: Composants (5/5)

❑ Les données traitées peuvent être de plusieurs sources (HDFS, Local, SGBDR,..).



❑ Spark offre plusieurs connecteurs pour accéder aux sources des données traitées.



Présentation d'Apache Spark: Spark VS Hadoop (1/7)

❑ **Finalité et composants**

- Hadoop est principalement conçu pour le stockage distribué et le traitement par lots de données massives. Ses composants principaux incluent HDFS pour le stockage et MapReduce pour le traitement.

- Spark est conçu pour le traitement des données par lots et en temps réel. Son abstraction principale est le **Resilient Distributed Dataset** (**RDD**), qui est une collection distribuée de données pouvant être traitées en parallèle sur un cluster.

Présentation d'Apache Spark: Spark VS Hadoop (2/7)

❑ **Paradigme de traitement**

- Le principal paradigme de traitement de Hadoop est **MapReduce**, qui consiste à décomposer un calcul en deux opérations **Map** et **Reduce**.

- Spark fournit un modèle de programmation plus flexible et plus expressif que MapReduce de Hadoop. Il prend en charge non seulement le traitement par lots, mais également les algorithmes itératifs, les requêtes interactives et le streaming en temps réel. Ce modèle est basé sur les **RDD**, les **transformations** et les **actions** qu'on peut appliquer aux RDD.

Présentation d'Apache Spark: Spark VS Hadoop (3/7)

❑ **Performance**

- MapReduce de Hadoop présente certaines limitations de performances en raison de son modèle d'exécution en deux étapes et de son stockage sur disque entre les étapes. Il est mieux adapté au traitement par lots qu'aux tâches interactives ou itératives.
- Le traitement en **mémoire** et le **moteur d'exécution optimisé** de Spark conduisent à des temps de traitement beaucoup plus rapides par rapport à MapReduce, surtout pour les charges de travail itératives et interactives.

Spark peut exécuter une application dans un cluster Hadoop, il est:

- jusqu'à 100 fois plus rapide en mémoire,
- Et 10 fois plus rapide lors de l'exécution sur le disque.

Présentation d'Apache Spark: Spark VS Hadoop (4/7)

❑ **Abstractions du traitement des données**

- L'écosystème Hadoop dispose d'outils tels que **Hive**, **Pig** et **HBase** qui fournissent des **abstractions** pour travailler avec les données massives, mais ces outils ne sont pas aussi optimisés pour le traitement en temps réel ou itératif.
- Spark propose des API de haut niveau telles que **DataFrames** et **Datasets**, qui offrent des opportunités d'**optimisation** et de meilleures performances. Il dispose également de bibliothèques pour l'apprentissage automatique (**MLlib**), le traitement de graphes (**GraphX**) et le **streaming** en temps réel.

Présentation d'Apache Spark: Spark VS Hadoop (6/7)

❑ **Facilité d'utilisation**

- Dans Hadoop, le **développement et le débogage des jobs MapReduce peuvent être complexes** et le cycle de développement peut être relativement lent.
- Les **API de Spark sont plus conviviales** et existent pour plusieurs langages de programmation tels que Python, Java et Scala. Cela rend le développement, les tests et le débogage plus rapides et plus intuitifs.

Présentation d'Apache Spark: Spark VS Hadoop (7/7)

❑ **Intégration**

- Spark peut s'exécuter sur Hadoop YARN, en utilisant HDFS pour le stockage, et il peut également lire les données de HBase, Hive et d'autres composants de Hadoop.

- Cependant, Spark ne s'appuie pas sur MapReduce de Hadoop et peut être utilisé en mode autonome (***Standalone Cluster Manager***) ou avec d'autres gestionnaires de cluster (***Apache Mesos***, ***Kubernetes***, ***Amazon EMR***, ***Google Cloud Dataproc***)

Présentation d'Apache Spark: Avantages (1/4)

❑ **Facilité d'utilisation**

- Spark fournit aux ingénieurs de données et des data scientists un moteur d'exécution et une **API unifiés** uniques pour **divers cas d'utilisation** tels que le traitement de streaming, par lots ou interactif et avec plusieurs langages de programmation.
- Ces outils lui permettent de faire face facilement à divers scénarios tels que les processus **ETL**, **l'apprentissage automatique** ou le **traitement de graphe**.
- Spark propose également une centaine **d'opérateurs de transformation** de données et la notion de **dataframes** et **datasets** pour manipuler des données **semi-structurées**.

Présentation d'Apache Spark: Avantages (2/4)

❑ **Facilité d'utilisation** *(suite)*

- La vaste documentation disponible facilite le développement d'applications Spark.

- Le modèle de traitement Hadoop MapReduce a inspiré la création d'Apache Spark dont le modèle est conceptuellement simple : diviser un programme complexe en **tâches** plus petites, **répartir** chaque tâche sur plusieurs nœuds, **collecter** les résultats partiels et les **agréger** dans un résultat final.

Présentation d'Apache Spark: Avantages (3/4)

❑ **Rapidité**

➢ Spark a été conçu pour atteindre des performances supérieures tant en **mémoire** que sur **disque**. Par conséquent, les opérations Spark s'effectuent sur le disque lorsque les données ne tiennent pas en mémoire.

➢ Le 5 novembre 2014, Databricks a utilisé un cluster Spark de **206** nœuds EC2 (Elastic Compute Cloud de Amazon) pour trier **100 To**(*10^12 enregistrements*) en **23min**. Le précédent record du monde de **72min** avec un cluster **Hadoop/MapReduce** de **2 100** nœuds avait été établi par **Yahoo**

➔ Spark a trié les mêmes données **trois fois** plus rapidement avec **dix fois** moins de machines en plus les tris étaient sur disque (HDFS), sans utiliser le cache en mémoire de Spark.

Présentation d'Apache Spark: Avantages (4/4)

❑ **Evolutivité**

➢ Apache Spark est un framework open source destiné à fournir un traitement **parallèle de données massives sur un cluster de machines**. Ceci est accompli en **distribuant les charges de travail (tâches)** sur un cluster d'ordinateurs (nœuds).

➢ Ces tâches sont orchestrées par **un gestionnaire de cluster** comme Apache Mesos ou Hadoop YARN. Par conséquent, les ressources matérielles peuvent augmenter de manière linéaire avec chaque nouveau nœud ajouté.