HADOOP ET SPARK

GROUPE 11 : Florine COMLAN, Ramya HOUNTONDJI

PSB 2020

INTRODUCTION

Vous souhaitez savoir comment fonctionne Hadoop et son écosystème, en particulier Spark dans un projet Big Data ? Vous êtes au bon endroit ! D'après le constat des experts, des institutions publiques et privés, 90 % des données récoltées depuis le début de l'humanité ont été générées durant les 2 dernières années. Le marché qualifie aujourd'hui de « Big Data » cette explosion de données. Il devient donc important pour tout un chacun de comprendre les principes de base des outils de gestion et de traitement de ces données massives tels que Hadoop et Spark. Ce document se déclinera suivant le plan suivant:

I- Hadoop et le big data

II- Composant de base d'un cluster hadoop

- a. Hadoop Distributed File System
- b. MapReduce

III- Hadoop et son écosystème

IV- Hadoop vs Spark

V- Comment installer Hadoop avec Docker?

VI- Packages

- a. RHadoop
- b. SparkR

I- Hadoop et le big data

Avant Hadoop, l'approche stratégique utilisée par les entreprises pour gérer leurs données consistait à centraliser le stockage et le traitement des données sur un serveur central dans une architecture client/serveur. Ces données sont gérées dans le serveur par un SGBDR (type Oracle, SQL Server, BD 2, etc). Le serveur central, ici, est une machine très puissante, conçue sur mesure par des sociétés spécialistes de l'infrastructure informatique comme EMC, Dell, HP ou encore Lenovo. La croissance des données de l'entreprise était gérée par upsizing du serveur, c'est-à-dire par augmentation de la capacité physique de ses composants. Par exemple, l'augmentation de la mémoire, de 124 Go à 256 Go, l'augmentation de la fréquence du processeur, de quadri-core 3 Ghz à Quadri-core 5 GHz, ou l'augmentation de la capacité de stockage du disque dur de 500 Go à 2 To. La figure suivante illustre cette stratégie.

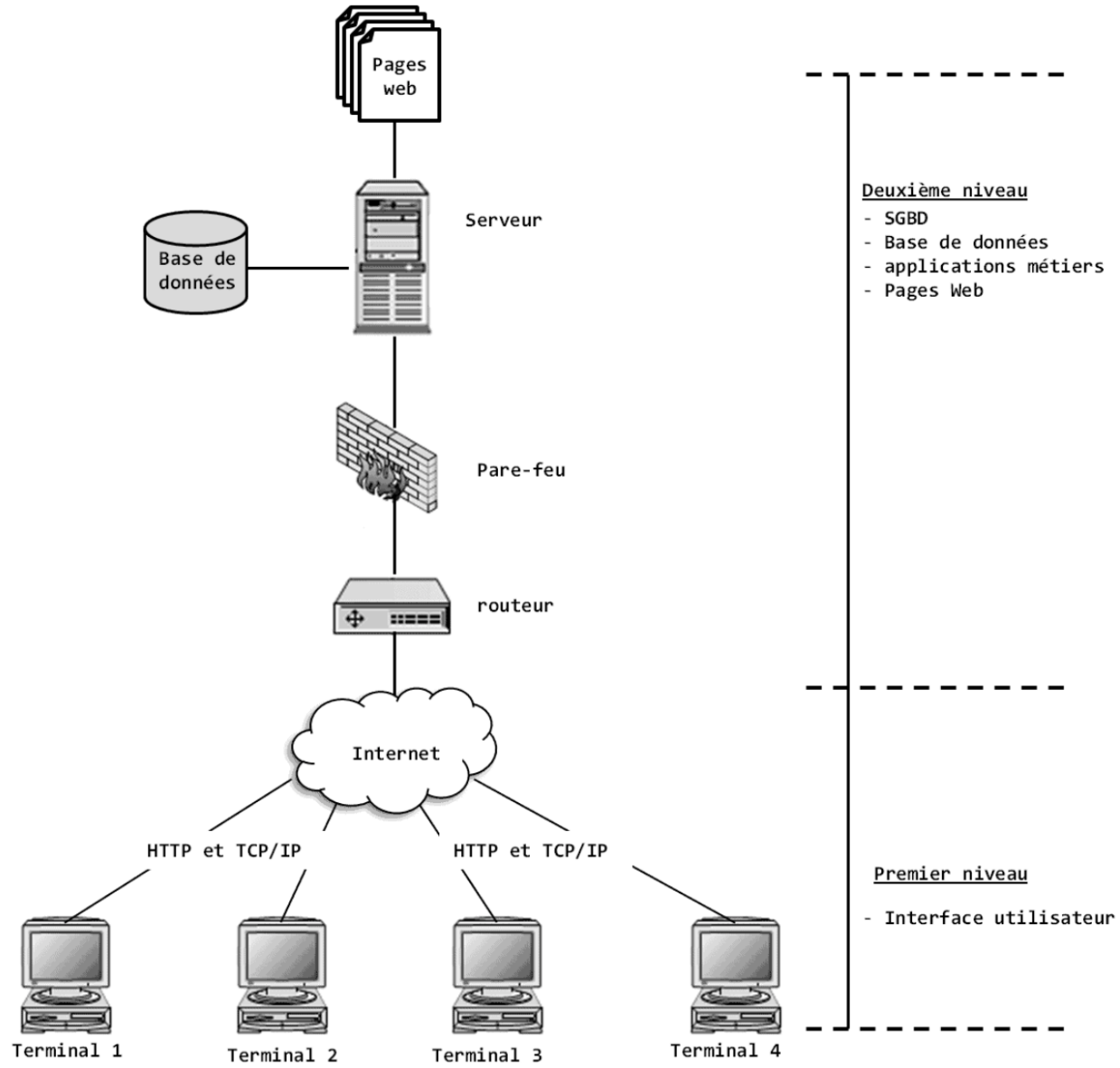


Figure: approche traditionnelle de gestion des données. Le stockage et le traitement des données sont centralisées dans un serveur.

Malheureusement, bien que de nombreuses entreprises gèrent encore leurs données selon cette stratégie, celle-ci pose plusieurs problèmes dans le contexte actuel :

- La centralisation du stockage et du traitement des données sur un serveur central crée une pression importante sur l'architecture informatique de l'entreprise, ce qui par effet domino augmente le temps de réponse des requêtes aux clients (la latence)
- L'upsizing qui est utilisé pour rendre le serveur central capable de s'adapter à l'augmentation du volume de données (on parle de scalabilité) est limité à la capacité maximale des composants informatiques.Par exemple, vous ne pouvez actuellement pas trouver sur le marché une barrette RAM de 500 Go. Pour atteindre cette capacité, il vous faut ajouter 8 barrettes de 64 Go, ce que la carte mère des serveurs ne prévoit pas toujours, en raison d'un nombre limité de slots

Google fait partie des premières entreprises qui ont très tôt ressenti ces faiblesses. Dans leur vision, avec la baisse des coûts d'ordinateurs tels que prédits par la loi de Moore, le futur du traitement informatique reposerait sur la constitution de Data Centers composés de plusieurs machines commodes (les clusters). Par ce point de vue, Google a introduit une nouvelle stratégie technologique qui va progressivement remplacer l'architecture client/serveur classique. En 2002, cette vision technologique paraissait ridicule, mais aujourd'hui, elle fait sens. En effet, l'approche proposée par Google consiste à distribuer le stockage des données et à paralléliser leur traitement sur plusieurs PC commodes organisées en cluster (on parle de nœuds). La figure ci-dessous illustre cette nouvelle stratégie. Hadoop est l'une des implémentations logicielles qui permet de mettre en œuvre cette approche.

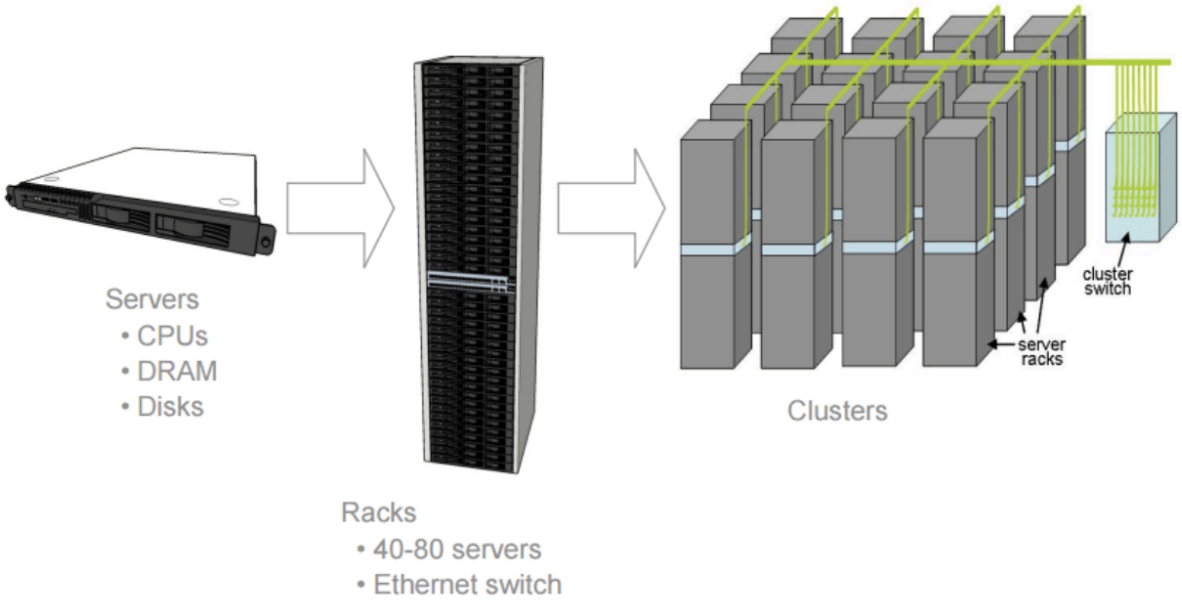


Figure: architecture distribuée – cluster computing. Le stockage des données est distribuée dans les noeuds d'un cluster et leur traitement y est parallélisé.

Ainsi, pour aborder des projets Big Data, d'un point de vue strictement architectural, la stratégie de distribution du stockage de données et de parallélisme de leur traitement sur un cluster est une bien meilleure stratégie que la stratégie d'architecture client/serveur classique. De plus, avec la baisse des coûts du matériel informatique, les coûts d'acquisition et d'évolution d'un cluster peuvent potentiellement revenir moins chers à terme que ceux d'un serveur central.

II- Composants de base d'un cluster hadoop

Avec le temps, un véritable écosystème technologique s'est développé autour d'Hadoop pour supporter la multiplicité de cas d'usage de valorisation de données et la multiplicité sectorielle d'industrie. Hadoop est quitté d'un logiciel « one-size-fits-all », c'est-à-dire comme un logiciel qui va fournir toutes les fonctionnalités

de tous les uses cases possibles du Big Data, à un véritable « framework », c'est-à-dire une plateforme de gestion de données sur laquelle peuvent être bâties des solutions spécifiques à des problématiques Big Data. Ainsi, lorsque vous lancer un projet en Big Data, gardez à l'esprit que l'acquisition d'Hadoop n'est que la première étape. Il faut déterminer (si elles existent) les solutions spécifiques qui peuvent vous aider à utiliser en levier la puissance d'Hadoop pour votre cas d'usage.

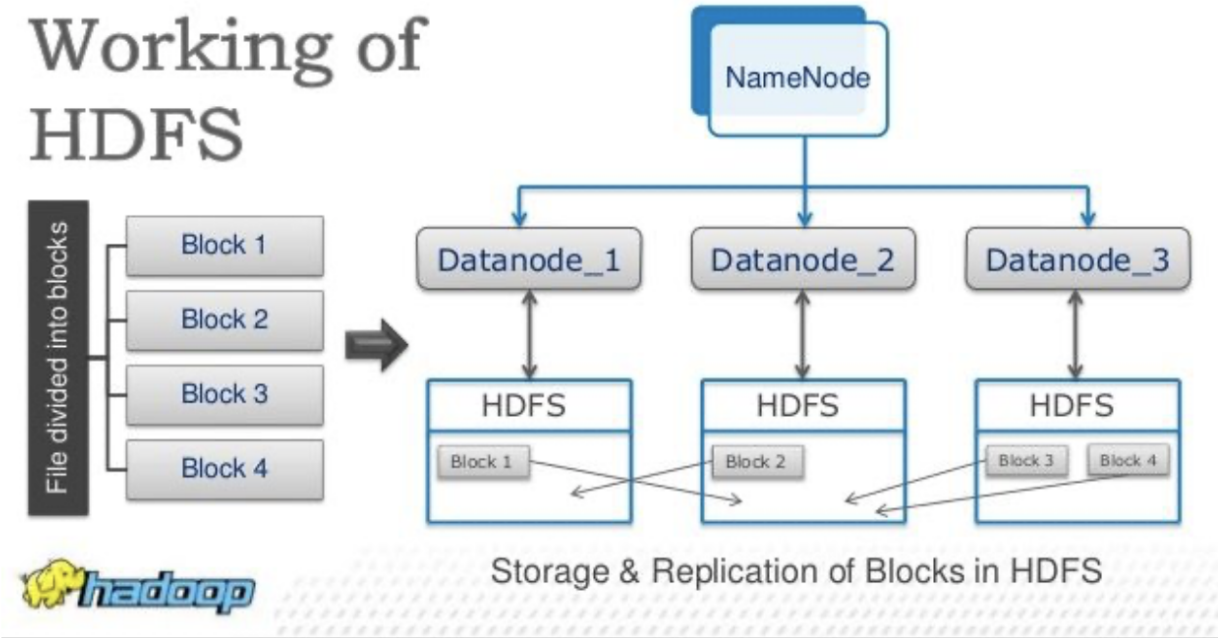
A ce jour, l'écosystème Hadoop est composé d'une centaines de technologies qu'on peut regrouper en 14 catégories selon leur segment de problématique. Nous y reviendrons plus bas. Mais ces technologies ont pour socle des composants de base tels que :

- Le système de fichier distribué, par exemple le HDFS d'Hadoop, qui gère le stockage distribué des données et fournit la tolérance aux pannes nécessaire lors de l'exploitation d'un cluster.
- le modèle de calcul, comme MapReduce, qui est la façon dont les données sont parallélisés dans les noeuds du cluster
- le gestionnaire de ressources, comme YARN, qui permet de faire tourner plusieurs moteurs de calcul dans le cluster et d'exploiter son potentiel à son maximum.

a. Hadoop Distributed File System

HDFS (Hadoop Distributed File System) est un système de fichier distribué permettant de stocker et de récupérer des fichiers en un temps record. Il s'agit de l'un des composants basiques du framework Hadoop Apache, et plus précisément de son système de stockage. De par sa capacité massive et sa fiabilité, HDFS est un système de stockage très adapté au Big Data. Parmi ses principales fonctionnalités, on compte la possibilité de stocker des terabytes, voire des petabytes de données.

Comment fonctionne t'il?



Le Hadoop Distributed File System repose sur une « architecture HDFS maître/esclave ». Chaque cluster comporte un Namenode individuel faisant office de serveur principal. Le Namenode se charge d'ouvrir, fermer, renommer les fichiers ou même les dossiers. Chaque nœud comporte un ou plusieurs Datanode, auquel est assignée la tâche de gérer le stockage associé au nœud. Les blocs sont cartographiés par le NameNode pour les DataNodes.

Notons que l'intégralité du HDFS est basée sur le langage de programmation Java.En réalité, le Namenode et le Datanode sont des codes de programmation Java pouvant être lancés sur des machines commodity

hardware.Ainsi, le Namenode de chaque cluster Hadoop centralise toute la gestion des dossiers et des fichiers afin d'éviter toute ambiguité.

La réplication de données est une partie essentielle du format HDFS. Comme le système est hébergé sur un commodity hardware, il est normal que les nœuds puissent tomber en panne sans crier gare. C'est pourquoi les données sont stockées de façon redondante, sous la forme d'une séquence de blocs. L'utilisateur peut facilement configurer la taille des blocs et le facteur de réplication. Les blocs de fichiers sont répliqués de façon à assurer la tolérance aux erreurs.

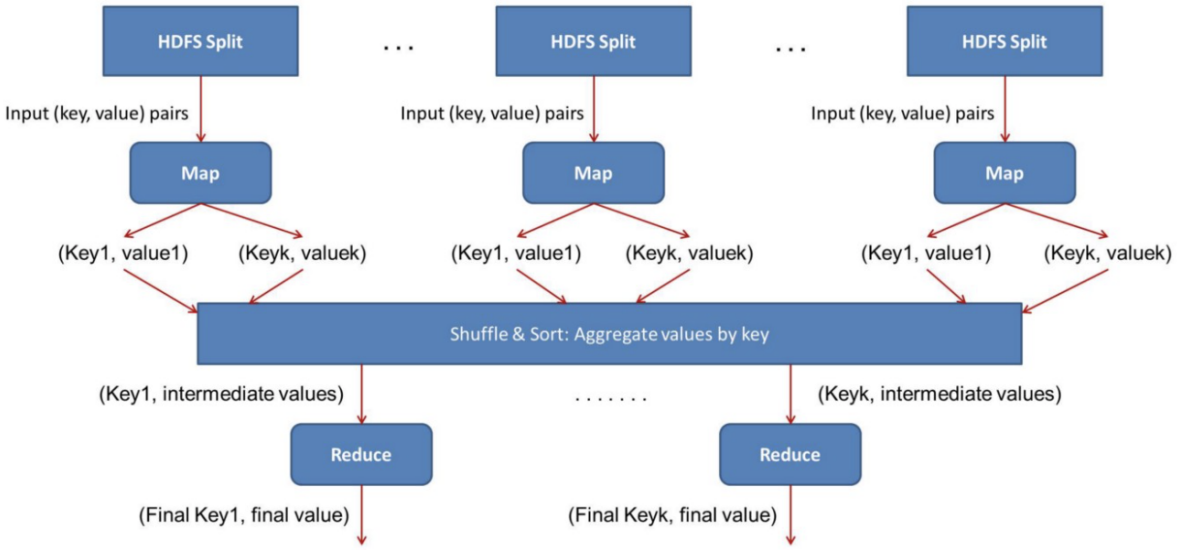
b. MapReduce

MapReduce est un modèle de programmation créé par Google pour le traitement et la génération de larges ensembles de données sur des clusters d'ordinateurs. Il s'agit d'un composant central du Framework logiciel Apache Hadoop, qui permet le traitement résilient et distribué d'ensembles de données non structurées massifs sur des clusters d'ordinateurs, au sein desquels chaque nœud possède son propre espace de stockage.

Chaque algorithme de traitement de données basé sur MapReduce ou encore job mapReduce doit contenir deux types de programmes :

- Les Mappers : Ce sont des fonctions qui reçoivent en entrée les données à traiter et produisent pour chaque ligne de données, un résultat sous forme d'une paire (clé,valeur).
- Les Reducers : Ces fonctions reçoivent les résultats des mappers et les réduisent au moyen de diverses sortes d'opérations afin d'en obtenir un résultat plus concis correspondant à celui attendu par l'utilisateur.

Voici en image, un aperçu des étapes d'un job MapReduce :



Architecture d'une porgramme MapReduce | Source : Data-flair

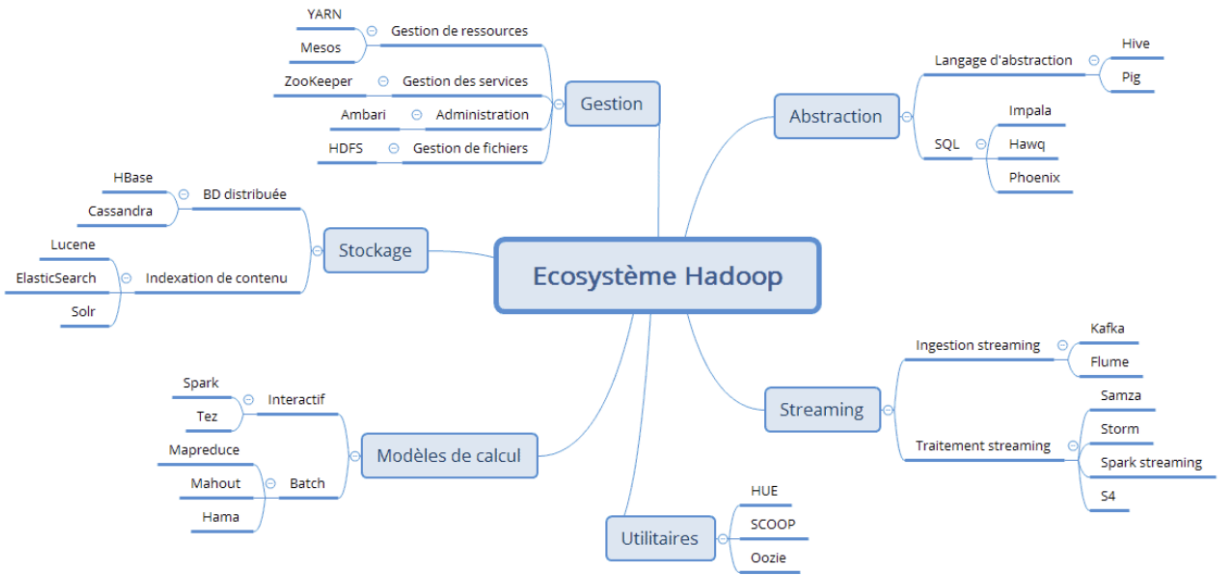
Supposons que vous ayez vos données stockées dans un cluster HDFS sur lequel vous exécutez un job MapReduce. L'exécution d'un programme se déroulera selon les étapes suivantes :

- La phase d'input : Les données HDFS sont lues et transmisses bloc par bloc chacune à un processus map séparé qui travaille uniquement sur les données qui lui sont assignées.
- La phase Map : Pendant cette phase, les instances de notre fonction map (que nous avons définie) s'exécute sur les blocs de données parallèllement et produit 0 ou plusieurs paires de (clé, valeur) appelées

clés intermédiaires. Nous verrons un peu plus bas à quoi correspondent ces clés, ces valeurs mais aussi le contenu d'un programme MapReduce.

- La phase combiner (optionnelle) : Écrire une fonction combiner n'est pas indispensable dans un programme MapReduce mais elle est utilisée pour réduire la taille des résultats du Mapper avant qu'ils soient transmis aux reducers.
- La phase Shuffle and Sort : Cette étape est automatiquement réalisée par MapReduce sans besoin qu'une fonction soit définie par l'utilisateur. Ici, MapReduce regroupe ensemble les résultats des différents mappers puis les trie par ordre croissant de leurs clés avant de les assigner, groupe par groupe à différents reducers.
- La phase Reducer : Un reducer est une fonction définie par l'utilisateur (tout comme un mappr) qui combine les groupes de paires (clé,valeur) en utilisant plusieurs techniques de filtrage,de sommage ou tout autre type d'opérations de groupage. Le reducer renvoie un set généralement très concis de paires (clé,valeur) à la dernière phase.
- La phase d'Output : Dans cette phase, les résultats du reducer sont lus et affichés à l'utilisateur ou (plus fréquemment) écrits dans un fichier.

III- Hadoop et son écosystème

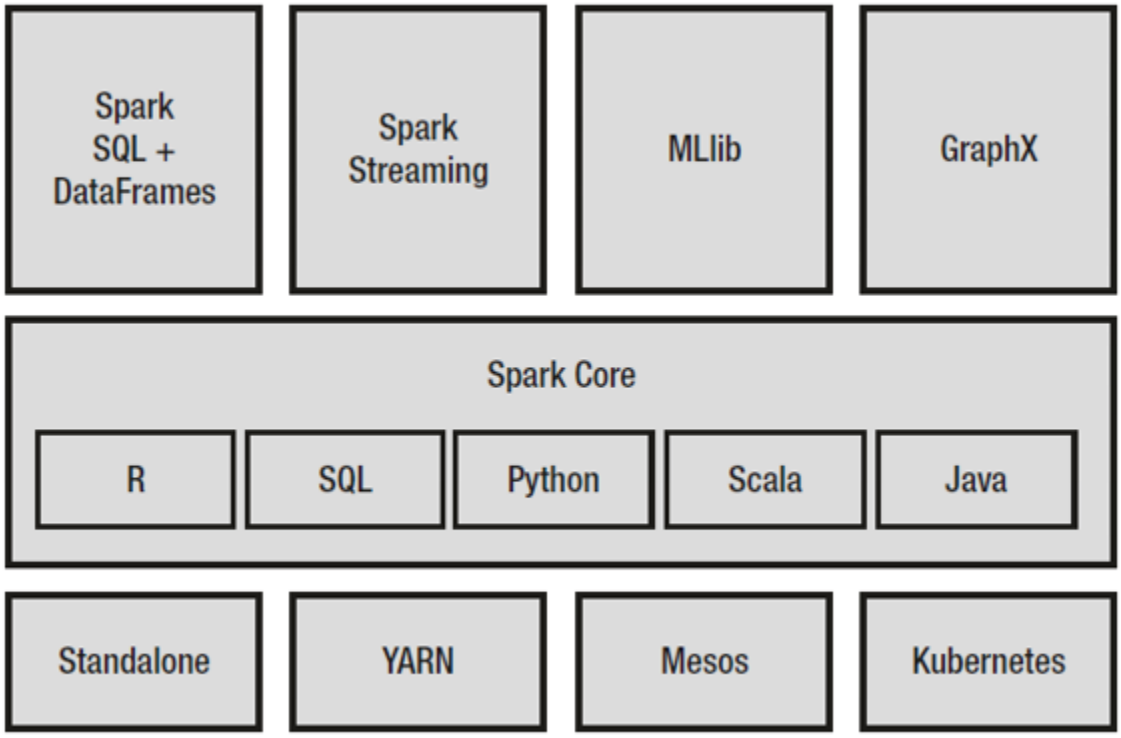


IV- Hadoop vs Spark

- Qu'est ce que Spark

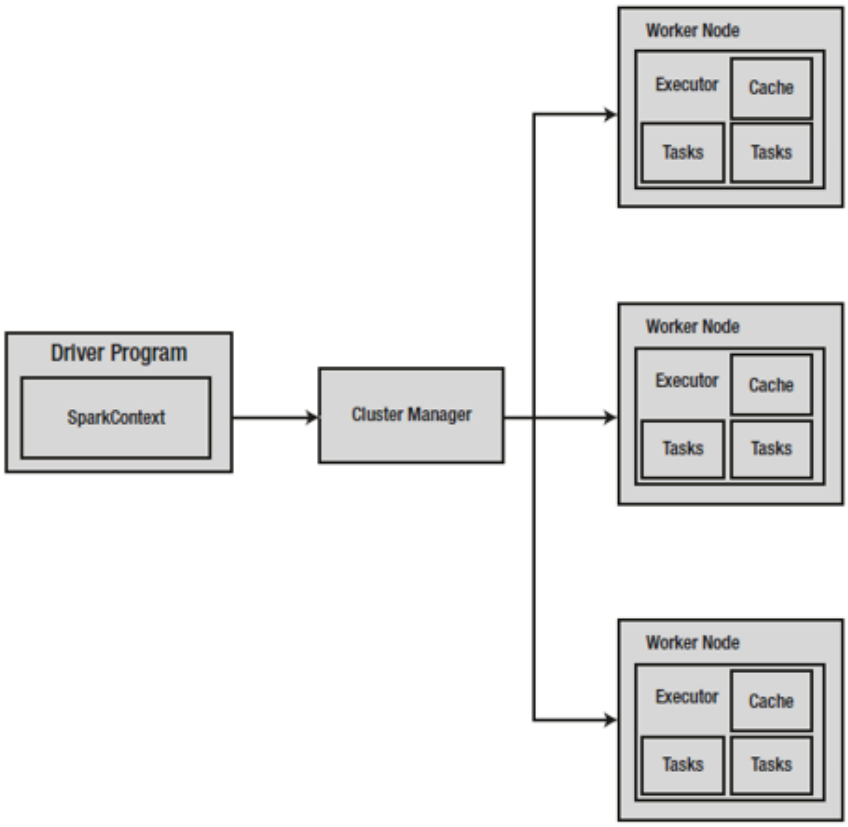
Spark est un moteur de traitement parallèle de données open source permettant d'effectuer des analyses de grandes envergures par le biais de machines en clusters. Il a été développé pour répondre aux limites de cadre de traitement. L'objectif était de créer un outil plus rapide et plus généralisé, qui pourrait être utilisé dans le cadre de traitrement multi-pass. Un autre avantage d' apache Spark est sa généralité. Il fait à la fois office de moteur de requêtes SQL, de logiciel de traitement de données en flux (Spark Streaming), et de système de traitement par graphes (GraphX). Apache Spark regroupe aussi une grande quantité de bibliothèques

d'algorithmes MLib pour le Machine Learning. Ces bibliothèques peuvent être combinées en toute simplicité au sein de la même application. Les tâches Spark peuvent s'exécuter beaucoup plus rapidement que les tâches MapReduce équivalentes en raison de leur grande capacités in memory et grâce au moteur d'exécution avancé DAG (graphique acyclique dirigé). Spark a été écrit en Scala.



- Architecture de Sparks

Les Cluster Manager gèrent et allouent les ressources du cluster. Spark prend en charge l'autonomie gestionnaire de cluster fourni qu'il fourni lui-même (planificateur autonome), YARN, Mesos et Kubernetes. À un niveau élevé, Spark répartit l'exécution des tâches des applications Spark sur l'ensemble du nœuds de cluster. Chaque application Spark a un objet SparkContext dans son programme pilote. Le SparkContext représente une connexion au cluster Manager, qui fournit des ressources informatiques aux applications Spark. En se connectant au cluster, Spark acquiert des exécuteurs sur les nœuds de calcul. Il envoie alors votre le code d'application aux exécuteurs. Une application exécute généralement un ou plusieurs travaux en réponse à une action Spark. Chaque Job est ensuite divisé par Spark en plus petits graphe acyclique (DAG) d'étapes ou de tâches. Chaque tâche est ensuite distribuée et envoyée aux exécuteurs testamentaires à travers les nœuds de travail pour l'exécution.



- Hadoop ou Sparks

Si les deux outils sont parfois considérés comme des concurrents, il est souvent admis qu'ils fonctionnent encore mieux quand ils sont ensemble. Voici un aperçu de leurs caractéristiques et de leurs différences.

Tous deux sont des frameworks big data, mais ils n'ont pas exactement le même usage. Hadoop est essentiellement une infrastructure de données distribuées : ce framework Java libre distribue les grandes quantités de données collectées à travers plusieurs nœuds (un cluster de serveurs x86), et il n'est donc pas nécessaire d'acquérir et de maintenir un hardware spécifique et coûteux. Hadoop est également capable d'indexer et de suivre ces données big data, ce qui facilite grandement leur traitement et leur analyse par rapport à ce qui était possible auparavant. Comparativement, Spark sait travailler avec des données distribuées. Mais il ne sait pas faire du stockage distribué. Il a donc besoin de s'appuyer sur un système de stockage distribué.

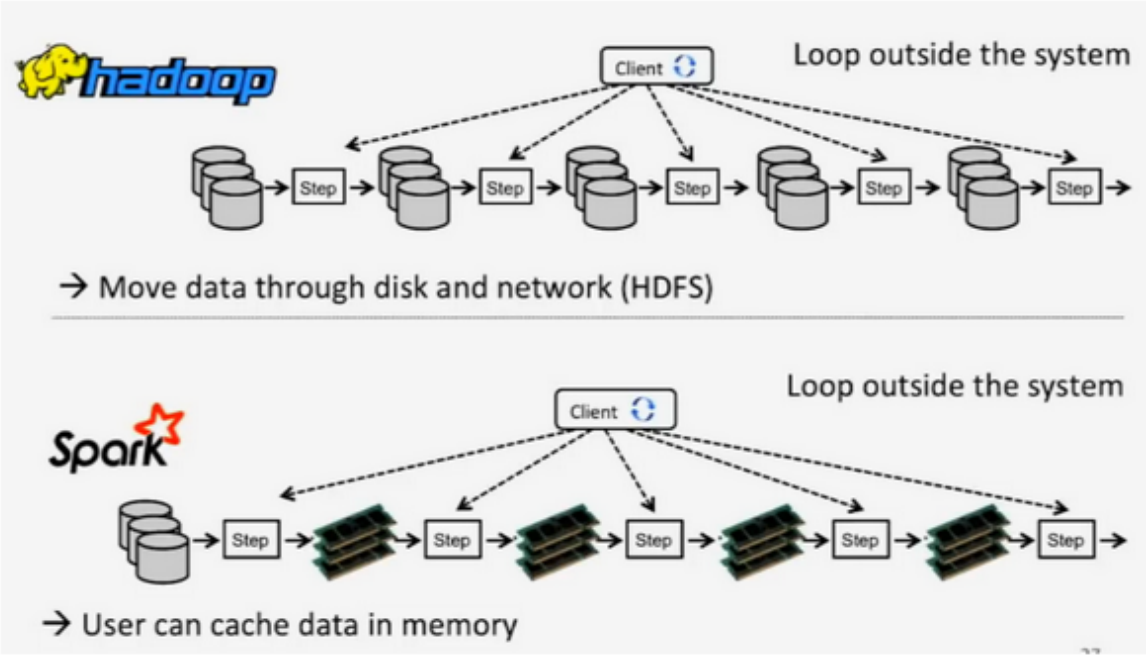
Hadoop comprend un composant de stockage, connu sous le nom de HDFS (Hadoop Distributed File System), et un outil de traitement appelé MapReduce. De fait, il n'est pas nécessaire de faire appel à Spark pour traiter ses données Hadoop. Et inversement, il est possible d'utiliser Spark sans faire intervenir Hadoop. Spark n'a pas de système de gestion de fichiers propre, ce qui veut dire qu'il faut lui associer un système de fichiers: soit HDFS, soit celui d'une autre plate-forme de données dans le cloud(S3..).

Spark est beaucoup plus rapide que Hadoop car la méthode utilisée par Spark pour traiter les données fait qu'il est beaucoup plus rapide que MapReduce. Alors que MapReduce fonctionne en étapes, Spark peut travailler sur la totalité des données en une seule fois. « La séquence de travail de MapReduce ressemble à ceci : il lit les données au niveau du cluster, il exécute une opération, il écrit les résultats au niveau du cluster, il lit à nouveau les données mises à jour au niveau du cluster, il exécute l'opération suivante, il écrit les nouveaux résultats au niveau du cluster, etc. », explique Kirk Borne, spécialiste des données chez Booz Allen Hamilton, un conseiller en gestion basé en Virginie. Au contraire, Spark exécute la totalité des

opérations d'analyse de données en mémoire et en temps quasi réel : « Spark lit les données au niveau du cluster, effectue toutes les opérations d'analyses nécessaires, écrit les résultats au niveau du cluster, et c'est tout. Cela le rend donc jusqu'à 10 fois plus rapide que MapReduce pour le traitement en lots et jusqu'à 100 fois plus rapide pour effectuer l'analyse en mémoire.

Il faut également noter que l'utilité de l'un ou de l'autre dépendra du besoin. Le mode de fonctionnement de MapReduce peut être suffisant si les besoins opérationnels et les besoins de reporting sont essentiellement statiques et s'il est possible d'attendre la fin du traitement des lots. Mais si l'on a besoin d'analyser des données en streaming, comme c'est le cas pour traiter des données remontées par capteurs dans une usine, ou si les applications nécessitent une succession d'opérations, il faudra probablement faire appel à Spark. C'est le cas de la plupart des algorithmes d'apprentissage machine qui ont besoin d'effectuer des opérations multiples. Spark est tout à fait adapté pour les campagnes de marketing en temps réel, les recommandations de produits en ligne, la cybersécurité et la surveillance des logs machine.

Au niveau des pannes, de la gestion des incidents et failles du système, par nature, Hadoop est performant car les données sont écrites sur le disque après chaque opération. Mais Spark offre la même résilience intégrée du fait que les objets de données sont stockés dans ce qu'on appelle des ensembles de données distribués résilients (RDD) répartis sur le cluster de données. Ces objets de données peuvent être stockés dans la mémoire ou sur les disques, et les ensembles RDD permettent une récupération complète après panne ou défaillance , fait encore remarquer Kirk Borne.



V- Comment installer Hadoop avec Docker?

Nous allons utiliser trois contenaires représentant respectivement un noeud maître (Namenode) et deux noeuds esclaves (Datanodes).

Etape 1: Installer Docker

Cliquez sur ce lien https://docs.docker.com/get-docker/ et suivez la procédure pour l'installer

Vérifiez la version installée avec :

```
docker --version
```

Etape 2: Télécharger l'image docker uploadée sur dockerhub

Cette image contient l'executable qui permet d'installer Hadoop (2.7.2), Spark (2.2.1), Kafka (2.11-1.0.2) et HBase (1.4.8).

- `docker pull liliasfaxi/spark-hadoop:hv-2.7.2`

Etape 3: Créer les trois contenaires à partir de l'image téléchargée. Pour cela:

- Créer un réseau qui permettra de relier les trois contenaires:

```
docker network create --driver=bridge hadoop
```

- Créer et lancer les trois contenaires (les instructions -p permettent de faire un mapping entre les ports de la machine hôte et ceux du contenaire):

```
docker run -itd --net=hadoop -p 50070:50070 -p 8088:8088 -p 7077:7077 -p 16010:16010 \
            --name hadoop-master --hostname hadoop-master \
            liliasfaxi/spark-hadoop:hv-2.7.2

docker run -itd -p 8040:8042 --net=hadoop \
        --name hadoop-slave1 --hostname hadoop-slave1 \
              liliasfaxi/spark-hadoop:hv-2.7.2
```

```
docker run -itd -p 8041:8042 --net=hadoop \
        --name hadoop-slave2 --hostname hadoop-slave2 \
              liliasfaxi/spark-hadoop:hv-2.7.2
```

Etape 4: Entrer dans le contenaire master pour commencer à l'utiliser.

```
docker exec -it hadoop-master bash
```

Le résultat de cette exécution sera le suivant:

```
root@hadoop-master:~#
```

Vous vous retrouverez dans le shell du namenode, et vous pourrez ainsi manipuler le cluster à votre guise. La première chose à faire, une fois dans le contenaire, est de lancer hadoop. Un script est fourni pour cela. Lancer ce script:

```
./start-hadoop.sh
```

Toutes les commandes interagissant avec le système Hadoop commencent par hadoop fs. Ensuite, les options rajoutées sont très largement inspirées des commandes Unix standard.

Etape 5: Ajouter des données dans HDFS

Pour ce faire, nous allons utiliser le fichier purchases.txt.Ce fichier se trouve sous le répertoire principal de votre machine master.

- Créer un répertoire dans HDFS, appelé input. Pour cela, taper:

```
hadoop fs -mkdir -p input
```

- Charger le fichier purchase.txt dans le répertoire input que vous avez créé:

```
hadoop fs -put purchases.txt input
```

- Pour afficher le contenu du répertoire input, la commande est:

```
hadoop fs -ls input
```

- Pour afficher les dernières lignes du fichier purchases:

```
hadoop fs -tail input/purchases.txt
```

Vous touverez dans le tableau ci-dessous les commandes les plus utilisées pour manipuler les fichiers dans HDFS

Instruction	Fonctionnalité
`hadoop fs -ls`	Afficher le contenu du répertoire racine
`hadoop fs -put file.txt`	Upload un fichier dans hadoop (à partir du répertoire courant linux)
`hadoop fs -get file.txt`	Download un fichier à partir de hadoop sur votre disque local
`hadoop fs -tail file.txt`	Lire les dernières lignes du fichier
`hadoop fs -cat file.txt`	Affiche tout le contenu du fichier
`hadoop fs -mv file.txt newfile.txt`	Renommer le fichier
`hadoop fs -rm newfile.txt`	Supprimer le fichier
`hadoop fs -mkdir myinput`	Créer un répertoire
`hadoop fs -cat file.txt | less`	Lire le fichier page par page

Etape 6: Visualiser

Hadoop offre plusieurs interfaces web pour pouvoir observer le comportement de ses différentes composantes.

Le port 50070: permet d'afficher les informations de votre namenode:

- `http://localhost:50070`

Le port 8088: permet d'afficher l'avancement et les résultats de vos Jobs (Map Reduce ou autre):

- `http://localhost:8088`

REFERENCES

- https://www.data-transitionnumerique.com/introduction-a-hadoop-et-l-ecosysteme-big-data/
- https://insatunisia.github.io/TP-BigData/tp1/index.html
- https://medium.com/takwimulab/mapreduce-le-framework-de-traitement-de-données-de-hadoop-f7624846440a
- https://www.lebigdata.fr/apache-spark-tout-savoir
- https://www.infoq.com/fr/articles/apache-spark-introduction/
- https://www.alphorm.com/tutoriel/formation-en-ligne-big-data-avec-apache-spark-initiation/tuto-video-comprendre-larchitecture-des-applications-de-spark