Cours
Mise en œuvre des Framework D'Intelligence Artificielle et Big data

Cycle Ingénieur
INDIA, Semestre 5

Pr. Abderrahim El Qadi
Département Mathématique Appliquée et Génie Informatique
ENSAM, Université Mohammed V de Rabat

A.U. 2023/2024

2eme partie

5. Hadoop Architecture
6. MapReduce & YARN
7. Hive
8. PIG & OOZIE
9. HBase
10. Data lake

Références

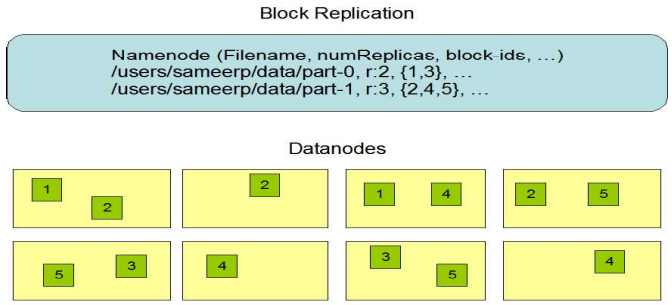
- Les bases de données NoSQL et le Big Data, Comprendre et mettre en œuvre. Rudi Bruchez. 2e edition
- Programming Hive. Data Warehouse and Query Language for Hadoop. Edward Capriolo, Dean Wampler, Jason Rutherglen - O'Reilly Media (2012)
- https://www.educba.com/hadoop-architecture
- https://www.educba.com/hadoop-commands
- https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
- https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

5. Hadoop Architecture

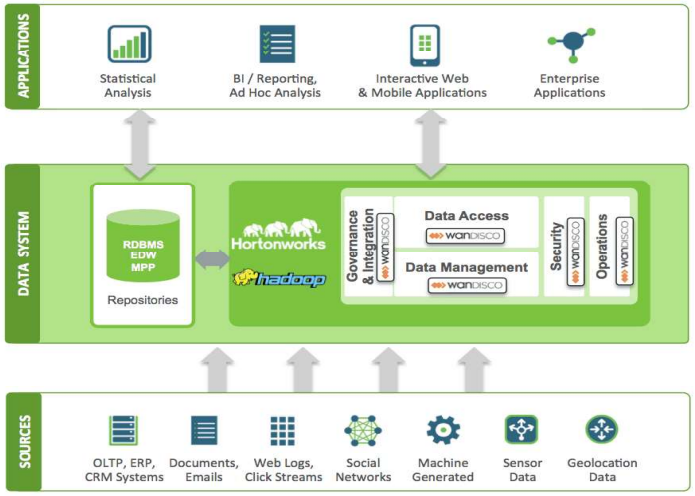
- **Big data → (3Vs) : Volume, Variété et Vélocité (fréquence).**
  - Problème du stockage et l'analyse des données.
    Solution 1 : Paralléliser les traitements en stockant sur plusieurs unités de disques durs.
    - Cela soulève forcément le problème de fiabilité des disques durs qui engendre une panne matérielle.

- **Hadoop** (**High-availability distributed object-oriented platform**) est un système distribué qui répond à ces problématiques.
  - Les machines en cluster et à faible coût peuvent lire l'ensemble de données en parallèle et offrent un débit beaucoup plus élevé.
  - De plus, c'est moins cher qu'un seul serveur haut de gamme.

- **Hadoop** a été l'une des premières technologies de Big Data open source.
  - Il est écrit en Java sous License Apache.
  - Il s'agit d'un écosystème logiciel qui permet aux entreprises de traiter d'énormes quantités de données en peu de temps.
  - C'est un "Framework", qui subdivise les données en plusieurs blocs plus petits et stocke chaque segment sur son propre nœud au sein du cluster.
  - Fournit un système d'analyse des données appelé **MapReduce** -> traitements sur des gros volumes de données.

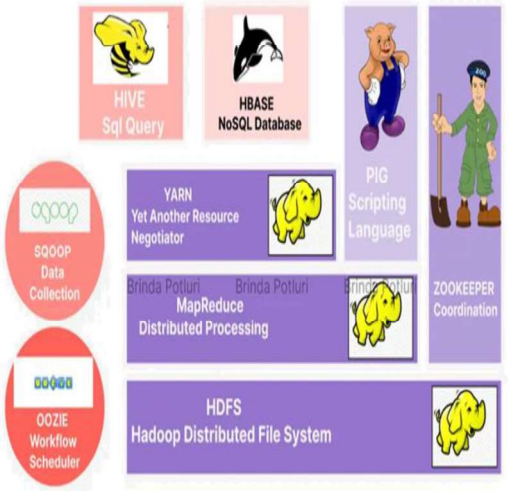


- **Hadoop dans l'entreprise**



- **Les distributions de Hadoop**

- **Ecosystème Hadoop**



HBase	une base de données NoSQL basée sur HDFS
Hive	une base de données relationnelle basée sur Hadoop, utilisable en SQL et accessible avec JDBC
Mahout	logiciel basé sur Hadoop fournissant un Framework et de nombreux algorithmes déjà implémentés pour effectuer du machine learning en se basant sur HDFS et MapReduce
Pig	outil de scripting basé sur Hadoop permettant de manipuler aisément de grandes quantités de données avec un langage proche du Python ou Bash
Oozie	une interface Web de gestion des jobs Hadoop pour les lancer et les planifier aisément en incluant les notions de dépendances de jobs à d'autres jobs

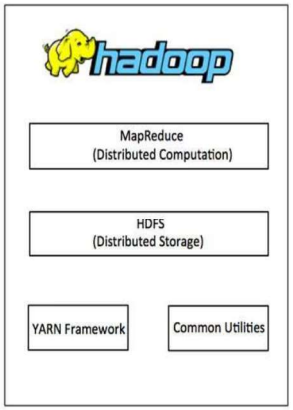
- **Evolution de Hadoop**

Hadoop 0.1 (2006)	- Il comprenait le système de fichiers distribués Hadoop (HDFS) et MapReduce, qui permettait aux développeurs d'écrire des programmes pour traiter des données en parallèle sur de nombreux nœuds.
Hadoop 1.0 (2011)	- Version stable de Hadoop. - Améliorations apportées à HDFS, telles que la prise en charge des ajouts de fichiers et des autorisations de fichiers, ainsi qu'un nouveau gestionnaire de ressources appelé YARN
Hadoop 2.0 (2013)	- Séparation les fonctions de gestion des ressources et de planification des tâches de MapReduce, permettant à d'autres frameworks de traitement de s'exécuter sur Hadoop. - Il a également introduit la prise en charge de l'exécution de Hadoop dans une configuration à haute disponibilité.

Hadoop 2.2 (2013)	- La prise en charge du codage d'effacement dans HDFS, ce qui a amélioré la durabilité des données et réduit les coûts de stockage.
Hadoop 3.0 (2017)	- La prise en charge de la conteneurisation, qui permettait à Hadoop de s'exécuter sur des plates-formes telles que Docker et Kubernetes. - Améliorations à HDFS, notamment la prise en charge d'un espace de noms hiérarchique et d'un nouvel algorithme de codage d'effacement appelé Reed-Solomon.
Hadoop 3.2 (2019)	- La prise en charge d'un codage d'effacement plus rapide et un nouvel algorithme de nettoyage de la mémoire qui réduisait la surcharge mémoire de Hadoop

-

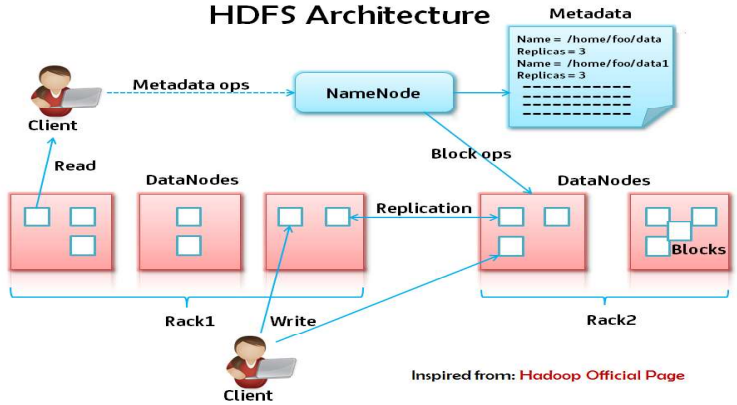
- Les composants de base d'un cluster Hadoop :
  - Le ***système de fichier distribué***, par exemple le **HDFS** d'Hadoop, un système de fichiers virtuel agrégeant le stockage de plusieurs machines d'un cluster ;
  - Le ***modèle de calcul***, comme **MapReduce,** un Framework logiciel en Java permettant de développer des programmes exécutables de manière distribués grâce à l'utilisation de l'algorithme MapReduce développé par Google ;
  - 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.



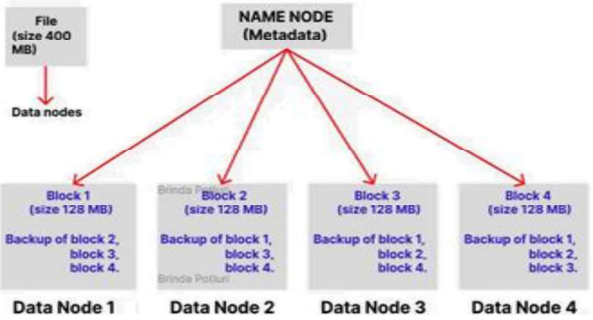
- .

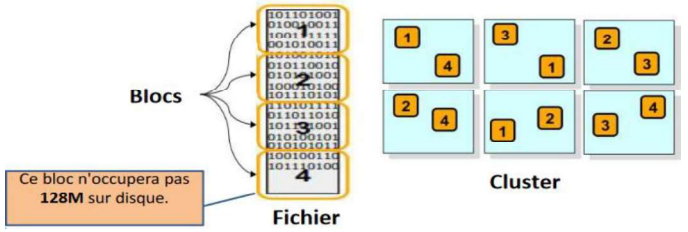
5.1. HDFS (Hadoop Distributed File System)

- HDFS : Système de fichiers conçu pour stocker des fichiers très volumineux dans un cluster d'ordinateurs pas chers.
  - HDFS stocke les données sur plusieurs nœuds qui forment un cluster



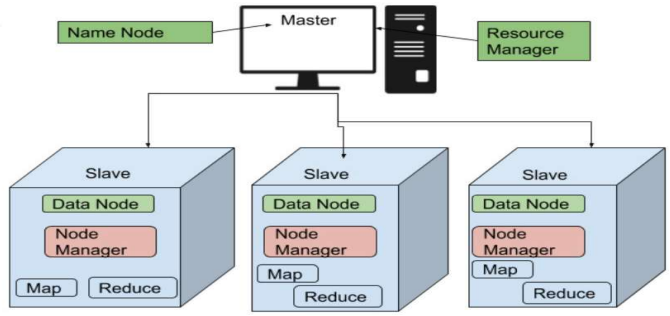
- HDFS réplique les données sur plusieurs nœuds afin d'éviter leur perte en cas de panne de certains nœuds : Fiabilité, Disponibilité
  - Chaque fichier est décomposé en plusieurs blocs qui seront répliqués sur les nœuds
  - La taille de bloc par défaut est de 128 Mo, mais elle est configurable.

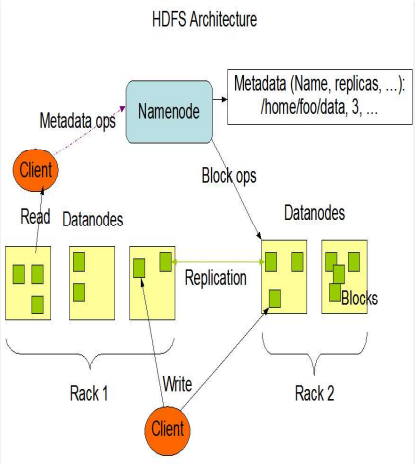




- Dans HDFS, la pratique par défaut consiste à répliquer chaque bloc de données trois fois.
  - Il est possible d'augmenter manuellement le nombre de réplicas à une valeur plus élevée.
- Dans HDFS, la fonction de gestion de la réplication assure la dispersion de ces réplicas sur de nombreux nœuds.
  - Elle assume également la responsabilité de la création ou de la suppression des réplicas en fonction des besoins de l'utilisateur.

- Un cluster HDFS se compose de deux types de nœuds : NameNode et DataNode.
  - Un **NameNode** gère l'espace de noms du système de fichiers. Il stocke toutes les métadonnées d'un fichier en mémoire. Par exemple, les noms de fichiers, les autorisations et les emplacements des blocs de fichiers.
  - Un **DataNode** stocke le contenu réel du fichier sous la forme de blocs de fichiers.

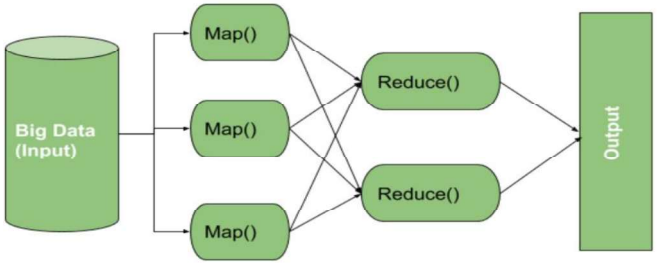


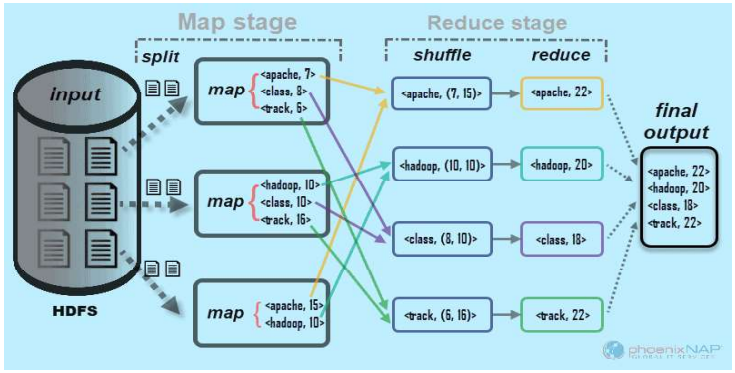


- Lorsqu'une application cliente veut lire un fichier, elle contacte d'abord un NameNode.
- Le NameNode répond avec les emplacements de tous les blocs qui composent ce fichier.
- Un emplacement de bloc identifie le DataNode qui contient les données pour ce bloc de fichier.
- Un client envoie alors directement une requête de lecture aux DataNodes pour chaque bloc de fichier.
- Un NameNode n'est pas impliqué dans le transfert de données réel d'un DataNode vers un client.
- De même, lorsqu'une application cliente souhaite écrire des données dans un fichier HDFS, elle contacte d'abord le NameNode et lui demande de créer une nouvelle entrée dans l'espace de noms HDFS.

5.2. MapReduce

– C'est une technique de programmation distribuée très utilisée dans le milieu NoSQL et qui vise à produire des requêtes distribuées.
  - Les blocs de construction de base d'une application MapReduce sont deux fonctions : mapper et réduire.
    - La fonction Map prend en entrée une paire clé-valeur et génère un ensemble de paires clé-valeur intermédiaires.
    - La fonction Reduce agrège ces valeurs et génère la valeur agrégée avec la clé intermédiaire qu'elle a reçu en entrée.



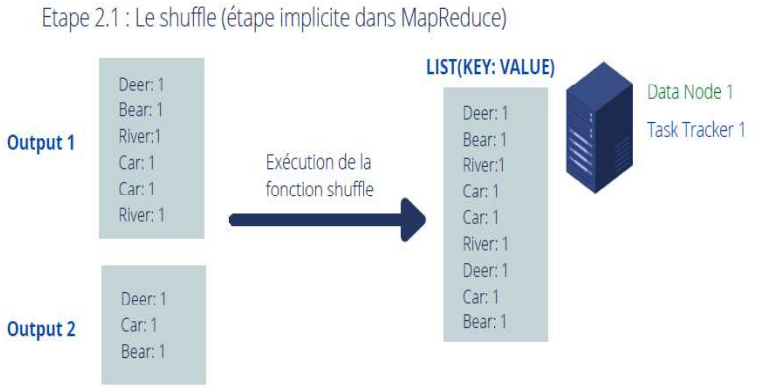


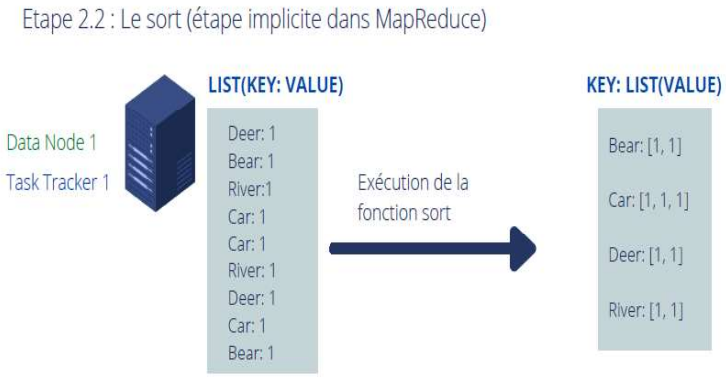
- L'étape de mapping peut être parallélisée en traitant l'application sur différents nœuds du système pour chaque couple clé-valeur.
- L'étape de réduction n'est pas parallélisée et ne peut être exécutée avant la fin de l'étape de mapping.

Exemple : Word Count

Compter la fréquence de chaque mot dans le fichier words.txt

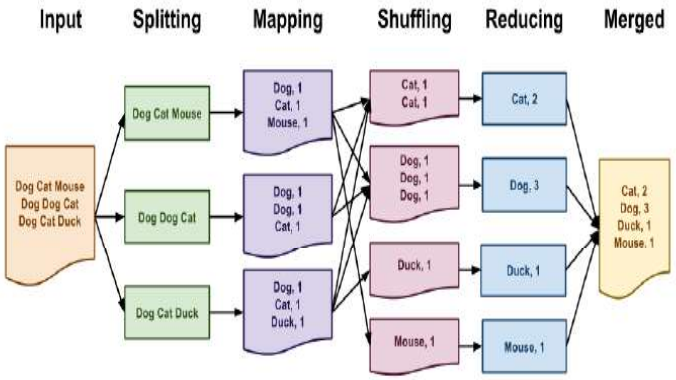

Etape 1: Le mapping
Block 1
Deer Bear River
Car Car River
Data Node 1
Task Tracker 1
Block 2
Deer Car Bear
Data Node 2
Task Tracker 2
Exécution de la fonction Map
KEY: VALUE
Deer: 1
Bear: 1
River:1
Car: 1
Car: 1
River: 1
Deer: 1
Car: 1
Bear: 1


Etape 2.1 : Le shuffle (étape implicite dans MapReduce)
Output 1
Deer: 1
Bear: 1
River:1
Car: 1
Car: 1
River: 1
Output 2
Deer: 1
Car: 1
Bear: 1
Exécution de la fonction shuffle
LIST(KEY: VALUE)
Deer: 1
Bear: 1
River:1
Car: 1
Car: 1
River: 1
Deer: 1
Car: 1
Bear: 1
Data Node 1
Task Tracker 1

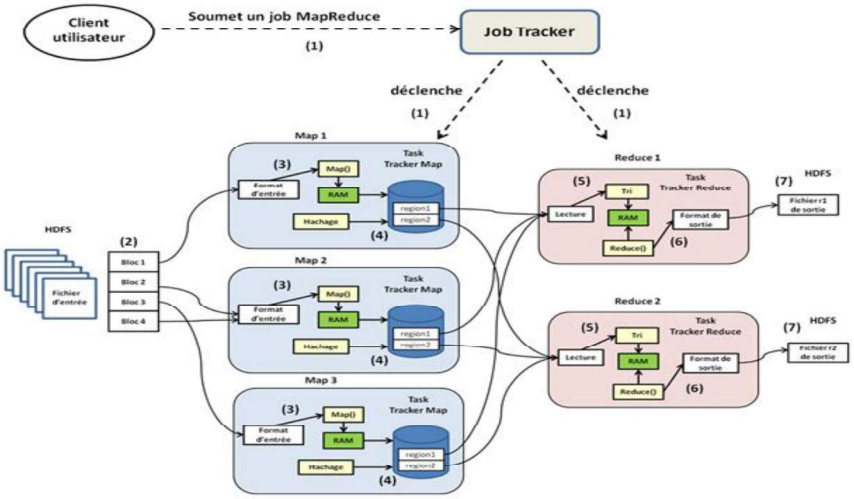

Etape 2.2 : Le sort (étape implicite dans MapReduce)
Data Node 1
Task Tracker 1
LIST(KEY: VALUE)
Deer: 1
Bear: 1
River:1
Car: 1
Car: 1
River: 1
Deer: 1
Car: 1
Bear: 1
Exécution de la fonction sort
KEY: LIST(VALUE)
Bear: [1, 1]
Car: [1, 1, 1]
Deer: [1, 1]
River: [1, 1]


Etape 3 : Le reduce
Data Node 1
Task Tracker 1
Bear: [1, 1]
Car: [1, 1, 1]
Deer: [1, 1]
River: [1, 1]
Exécution de la fonction reduce
KEY: LIST(VALUE)
Bear: 2
Car: 3
Deer: 2
River: 2

Processus complet d'un job MapReduce



- Le traitement MapReduce écrit par l'utilisateur s'appelle un **job MapReduce** et s'exécute en six étapes :



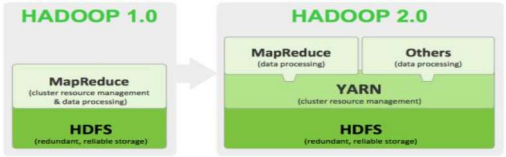
Etape	
1	L'utilisateur configure le Job MapReduce : • Écrit les deux fonctions Map, et Reduce ; spécifie le nombre de tâches Reduce (r), le format de lecture du fichier d'entrée, le format de sortie des r fichiers Reduce, la taille des blocs du fichier d'entrée et le facteur de réplication. • Déclenche l'exécution du job, le jobtracker démarre les r tasktrackers qui vont effectuer les r tâches Reduce que l'utilisateur a spécifiées ;
2	Le HDFS découpe le fichier d'entrée en M blocs de taille fixe (64 Mo) Le HDFS réplique ces blocs selon le facteur de réplication défini par l'utilisateur (trois par défaut) et les distribue de façon redondante dans des nœuds différents dans le cluster.
3	Le jobtracker déclenche M tasktrackers sur les M nœuds de données dans lesquels ont été répartis les M blocs du fichier d'entrée, pour exécuter les tâches Map, soit un tasktracker Map pour chaque bloc de fichier.

4	Dans chaque nœud, les paires de clés/valeurs sont sérialisées dans un fichier sur le disque dur local du nœud. • Ce fichier est partitionné en r régions par une fonction de hachage qui va assigner à chaque région une clé qui correspond à la tâche Reduce à laquelle elle a été assignée. • Les informations sur la localisation de ces régions sont transmises au jobtracker, qui fait suivre ces informations aux r tasktrackers qui vont effectuer les tâches Reduce ;
5	Lorsque les r tasktrackers Reduce sont notifiés des informations de localisation, • Ils utilisent des appels de procédures distantes (protocole RPC) pour lire depuis le disque dur des nœuds sur lesquels les tâches Map se sont exécutées, les régions des fichiers Map leur correspondant. • Ensuite, ils les trient par clé. • Les tasktrackers Reduce itèrent à travers toutes les données triées et pour chaque clé unique rencontrée, ils la passent avec sa valeur à la fonction Reduce écrite par l'utilisateur.

	• Les résultats du traitement de la fonction Reduce sont alors sérialisés dans le fichier ri (avec i l'indice de la tâche Reduce) selon le format de sortie spécifié par l'utilisateur. Cette fois-ci, les fichiers ne sont pas sérialisés dans le disque dur du nœud tasktracker, mais dans le HDFS, ceci pour des raisons de résilience (tolérance aux pannes) ;
6	• Le job s'achève là, à ce stade, les r fichiers Reduce sont disponibles et Hadoop applique en fonction de la demande de l'utilisateur, soit un « Print Écran », soit leur chargement dans un SGBD, soit alors leur passage comme fichiers d'entrée à un autre job MapReduce.

5.3. YARN (Yet Another Resource Negociator)

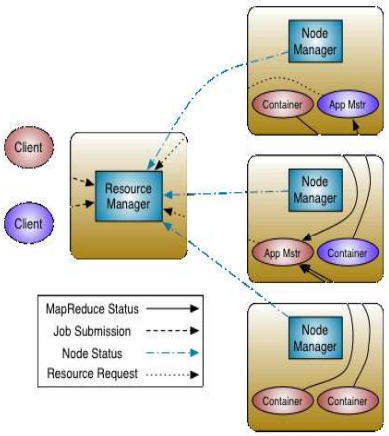
- Outil de gestion des ressources distribuées.
  Provient d'un découpage de la première version de Hadoop MapReduce en deux sous-couches :
  - L'une dédiée à la gestion de la puissance de calcul et de la répartition de la charge entre les machines d'un cluster (YARN)
  - L'autre dédiée à l'implémentation de l'algorithme MapReduce en utilisant cette première couche



- Yarn gère des ressources telles que la mémoire, le réseau, le disque et le processeur.
- Dans l'ensemble, Yarn permet au cluster Hadoop de s'attaquer à un plus grand nombre de charges de travail.

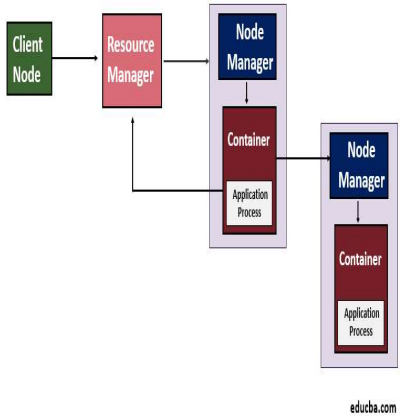
– L'idée fondamentale de YARN est de diviser les fonctionnalités de gestion des ressources et de planification/surveillance des tâches en démons distincts.

- Le ResourceManager et le NodeManager forment l'infrastructure de calcul des données.
  - ResourceManager est l'autorité ultime qui arbitre les ressources entre toutes les applications du système.
  - Le NodeManager est l'agent d'infrastructure par machine qui est responsable des conteneurs, de la surveillance de leur utilisation des ressources (processeur, mémoire, disque, réseau) et de la signaler au ResourceManager/Scheduler.
- L'ApplicationMaster par application est, en fait, une bibliothèque spécifique à l'infrastructure et est chargée de négocier les ressources du ResourceManager et de travailler avec le ou les NodeManager pour exécuter et surveiller les tâches.



- Étapes d'exécution d'une application dans YARN

1. L'utilisateur télécharge une application dans ResourceManager de Yarn.
2. Yarn alloue des ressources en fonction des besoins.
3. Le maître d'application de Yarn établit un lien avec le NodeManager associé, ce qui lui permet d'utiliser le conteneur.
4. NodeManager exécute le conteneur.
5. Yarn libère les ressources une fois qu'il a exécuté le conteneur et que l'application a fini de s'exécuter.



5.4. Hadoop: Modes de fonctionnement

- Hadoop possède trois modes d'exécution :
  - Mode local (standalone) : Hadoop fonctionne sur une seule machine et tout s'exécute dans la même JVM (Java Virtual Machine).

    En mode local le système de gestion de fichiers utilisé est celui du système hôte.

  - Mode pseudo-distribué : Hadoop fonctionne sur une seule machine, mais chacun des daemons principaux s'exécute dans sa propre JVM.
    Le système de fichier utilisé est HDFS dans ce mode.

  - Mode totalement distribué : c'est le mode d'exécution réel d'Hadoop.
    Il permet de faire fonctionner le système de fichiers distribué et les daemons sur un ensemble de machines différentes

5.5. Commandes de Hadoop

- Deux façons pour manipuler HDFS
  via l'API Java
  via les commandes depuis un terminal
- Utilisation du Hadoop avec un seul nœud pour tester quelques commandes HDFS.
  • Les clients HDFS utilisent la propriété fs.default.name ou fs.defaultFS de Hadoop pour déterminer l'URL de l'hôte et le port du NameNode. (fs.defaultFS = hdfs://localhost: 8020) avec le port HDFS par défaut, 8020.
  • Les propriétés et paramètres Hadoop sont dans des fichiers de configuration:
  /etc/hadoop/conf/core-site.xml
  /etc/hadoop/conf/hdfs-site.xml:
  facteur de réplication (dfs.replication), NameNode secondaire (dfs.namenode.secondary.http-address), …

hdfs dfs **version**	Affiche la version du Hadoop
hdfs dfs **-mkdir** /usr/local/firstdir?	Créer de nouveaux répertoires et prend le chemin d'accès à l'URI en paramètre
hdfs dfs **-ls** /usr/local/firstdir	afficher la liste du contenu d'un répertoire
hdfs dfs **-put** source_dir destination_dir	copier le contenu du système de fichiers local vers l'autre emplacement de DFS.
hdfs dfs -**copyFromLocal** local_src destination_dir	même que la commande put
hdfs dfs **-get** source_dir local_dir?	récupère tous les fichiers qui correspondent au répertoire source
hdfs dfs **-cat** dir_path	affiche le contenu du nom de fichier
hdfs dfs **-rm** dir_name	supprime les fichiers et le répertoire du chemin spécifié
hdfs dfs **-du** dir_name	affiche l'utilisation du disque pour tous les fichiers disponibles dans le répertoire
hdfs dfs **-df** -h	affiche l'espace libre
hdfs dfs **-count** dir_name	compte le nombre de répertoires et de fichiers.

5.6. Avantages et inconvénients du Hadoop

Avantages	Inconvénients
permet à l'utilisateur d'écrire et de tester rapidement des systèmes distribués	trop difficile à apprendre pour les développeurs débutants
Il est efficace et distribue automatiquement les données et le travail sur les machines et utilise à son tour le parallélisme sous-jacent des cœurs du processeur	Rendre le traitement des données en temps réel évolutif peut nécessiter d'investir dans une infrastructure supplémentaire
ne s'appuie pas sur le matériel pour fournir une tolérance aux pannes et une haute disponibilité	Le traitement des données en temps réel a une portée très limitée
la bibliothèque Hadoop elle-même a été conçue pour détecter et gérer les pannes au niveau de la couche application.	Il y a de sérieux problèmes de sécurité en raison de sa plate-forme open-source
Les serveurs peuvent être ajoutés ou supprimés du cluster de manière dynamique et Hadoop continue de fonctionner sans interruption	Les utilisateurs doivent avoir l'expertise nécessaire pour l'utiliser correctement
compatible sur toutes les plateformes puisqu'il est basé sur Java	