Université Constantine 2
جامعة قسنطينة 2

Big Data & NoSQL (BDNS)

– Chapitre 4–

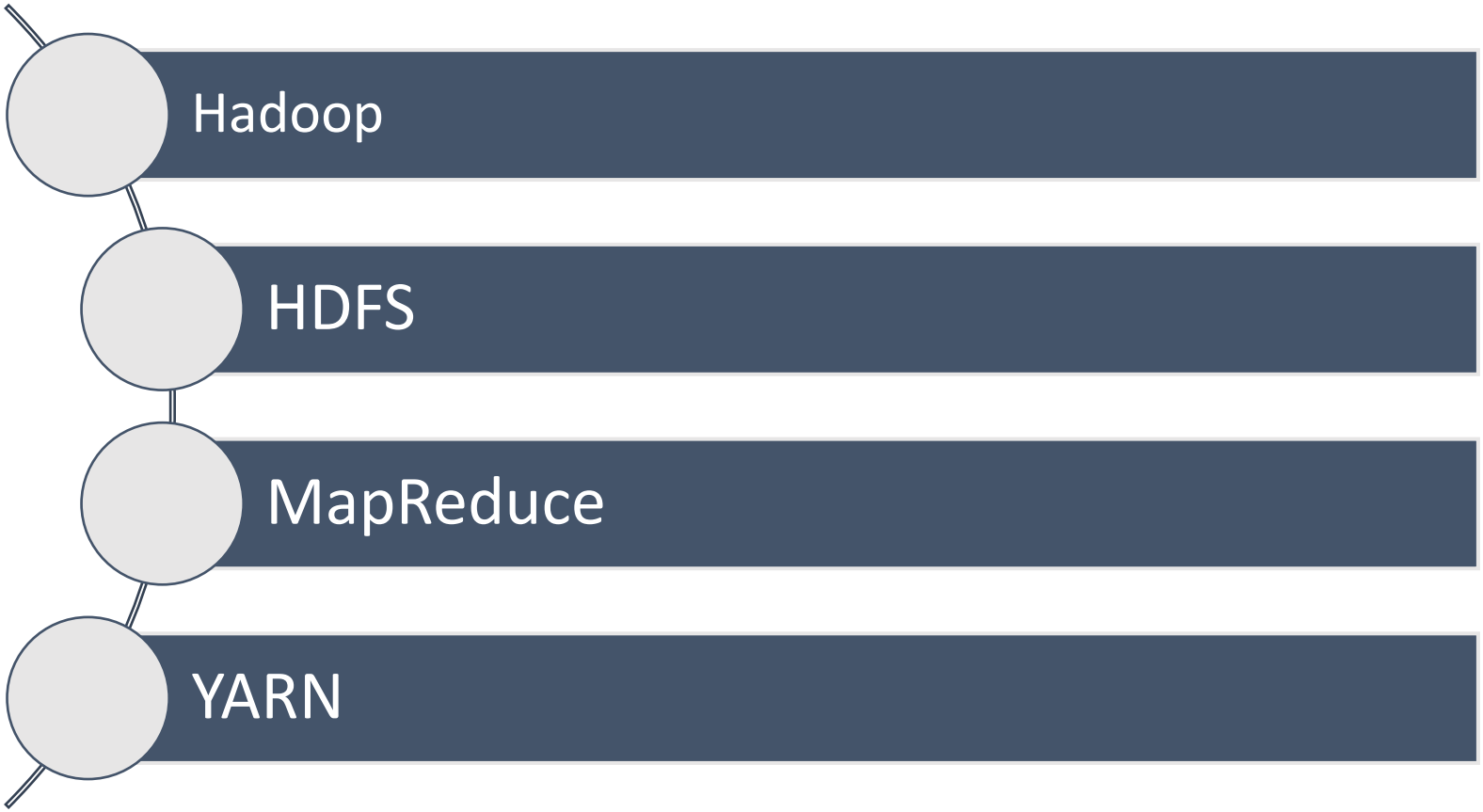
Hadoop , HDFS et MapReduce

Dr. GHEMMAZ W
Faculté NTIC
Wafa.ghemmaz@univ-constantine2.dz

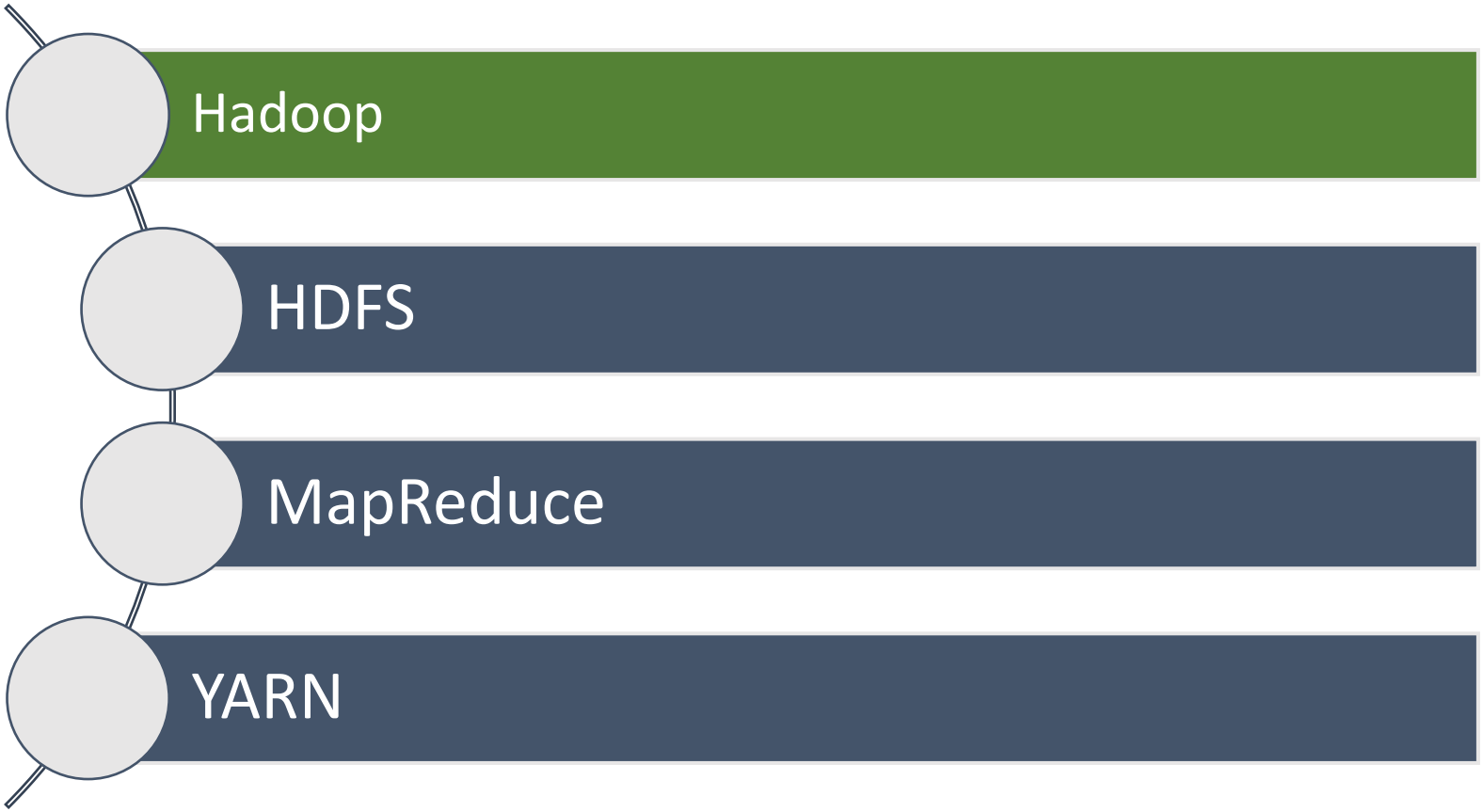
Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	TLSI	Master 1	SDSI

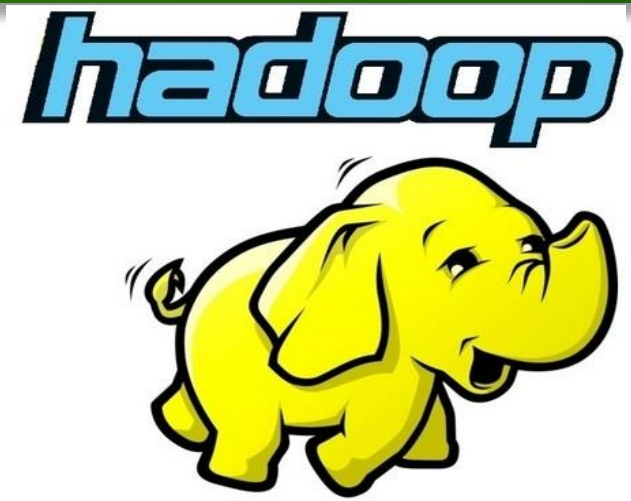
Plan



Plan



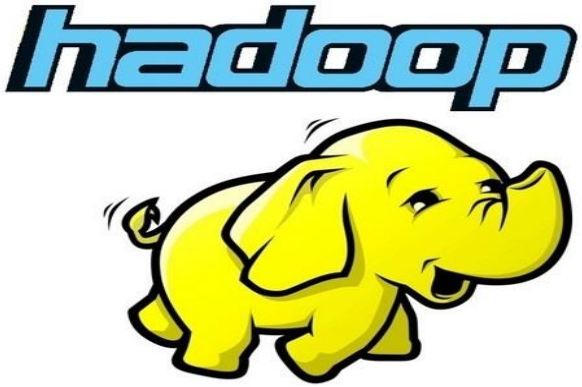
Hadoop



- Hadoop est un système de répartition des calculs qui exploite un système de fichiers particulier nommé HDFS (Hadoop Distributed File System)
- Hadoop est destiné à faciliter la création d'applications distribuées sur des milliers de nœuds et des pétaoctets de données.

Hadoop

Hadoop

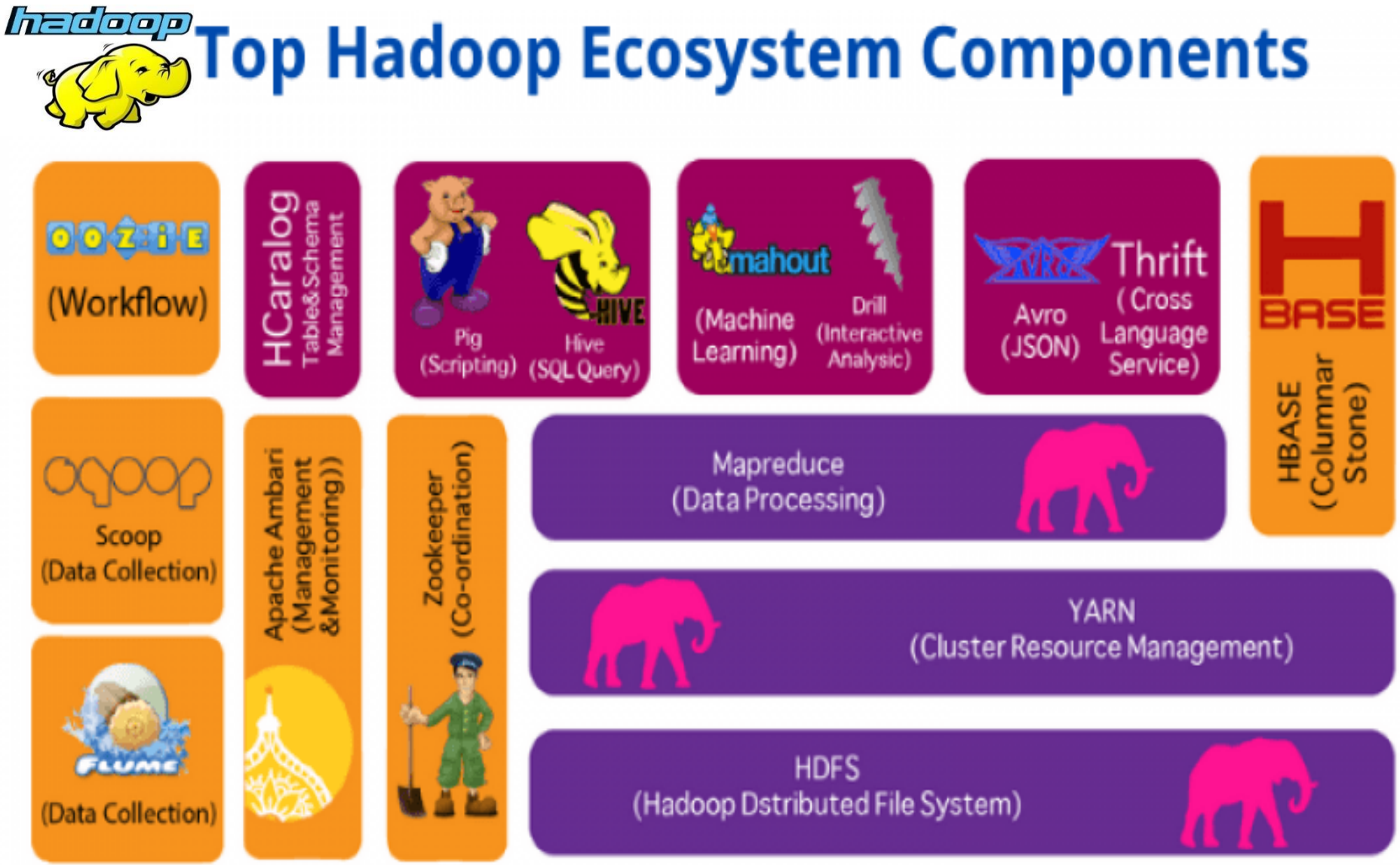


Le projet Hadoop consiste essentiellement en deux grandes parties:

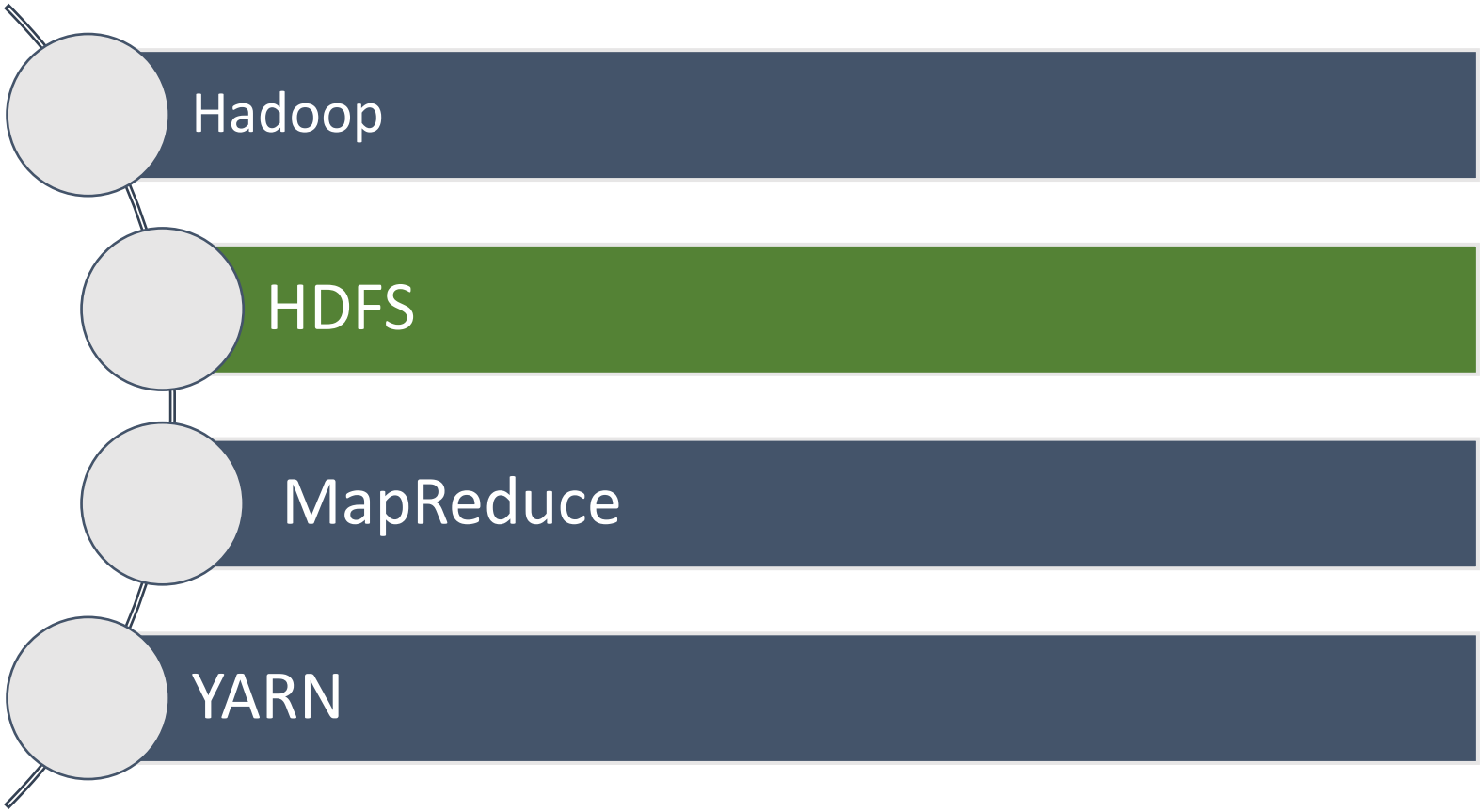
- Stockage des données : HDFS
- Traitement des données : MapReduce

Hadoop comprend de nombreuses briques logiciels (Ecosystème)

Hadoop


Top Hadoop Ecosystem Components
(Workflow)
HCaralog
Table&Schema Management
Pig
(Scripting)
Hive
(SQL Query)
mahout
(Machine Learning)
Drill
(Interactive Analysic)
Avro
(JSON)
Thrift
(Cross Language Service)
HBASE
(Columnar Stone)
Scoop
(Data Collection)
Apache Ambari
(Management &Monitoring))
Zookeeper
(Co-ordination)
Mapreduce
(Data Processing)
(Data Collection)
YARN
(Cluster Resource Management)
HDFS
(Hadoop Dstributed File System)

Plan



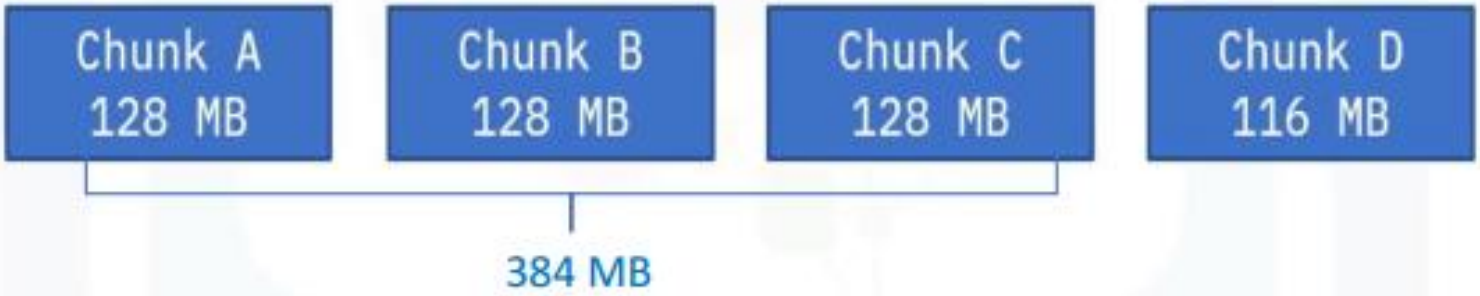
HDFS

- HDFS permet de stocker des fichiers très volumineux de manière distribuée.
- Les fichiers sont physiquement découpés en blocs d'octets de grande taille (par défaut 128 Mo) pour optimiser les temps de transfert et d'accès ;
- Ces blocs sont ensuite répartis sur plusieurs nœuds, permettant ainsi de traiter un même fichier en parallèle

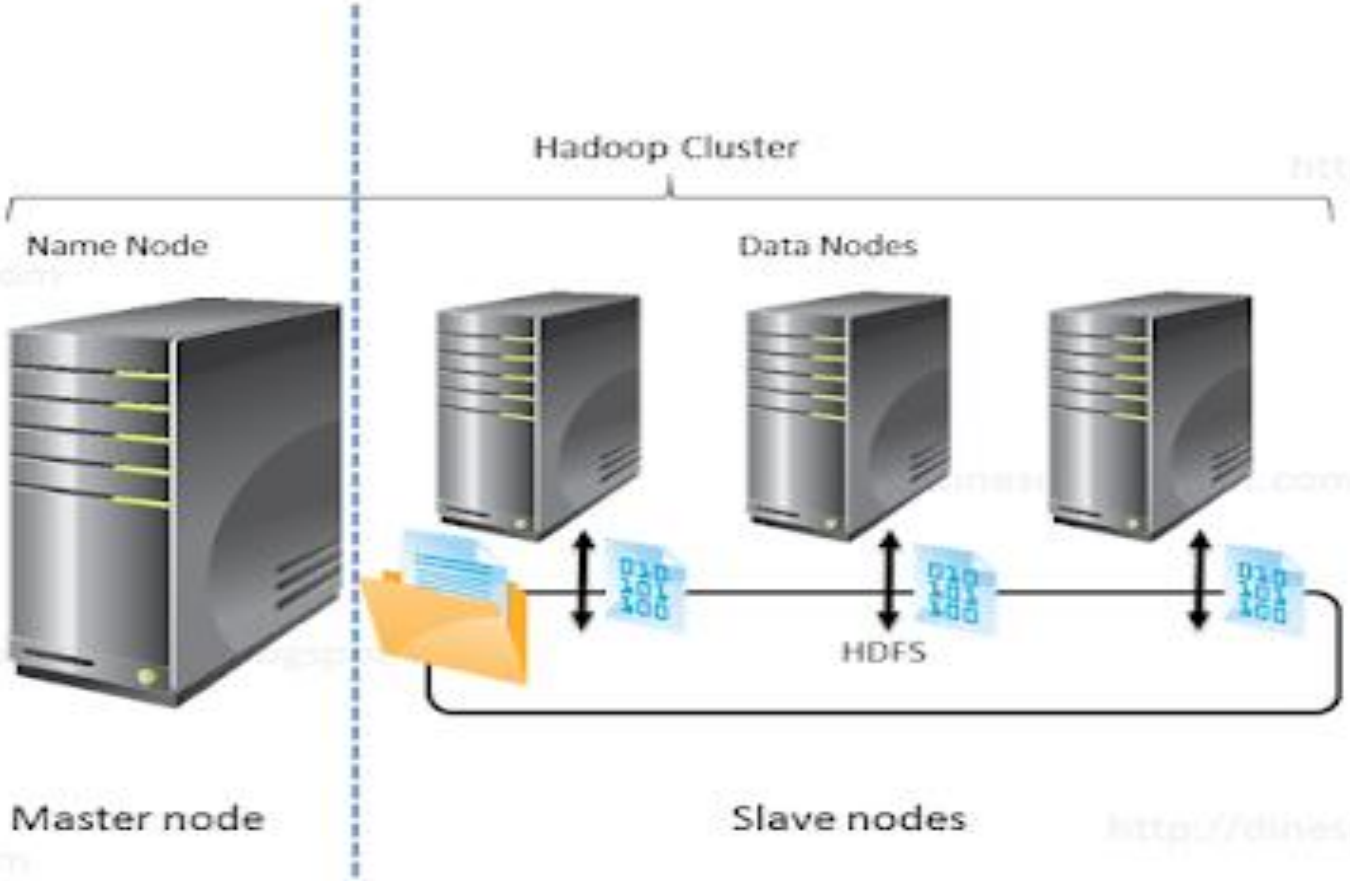
HDFS

Blocks

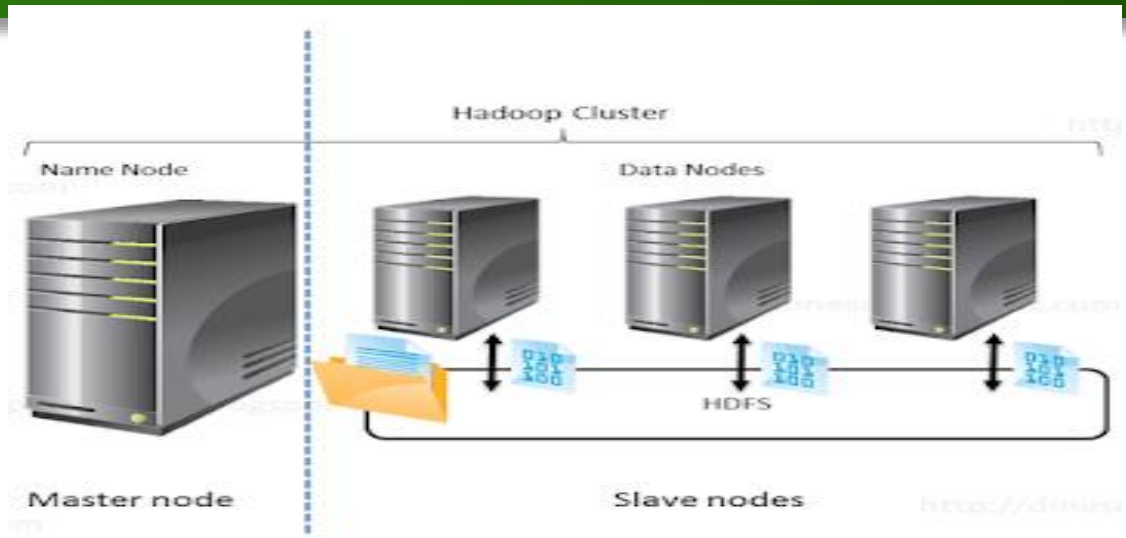
File size: 500 MB



HDFS: Architecture

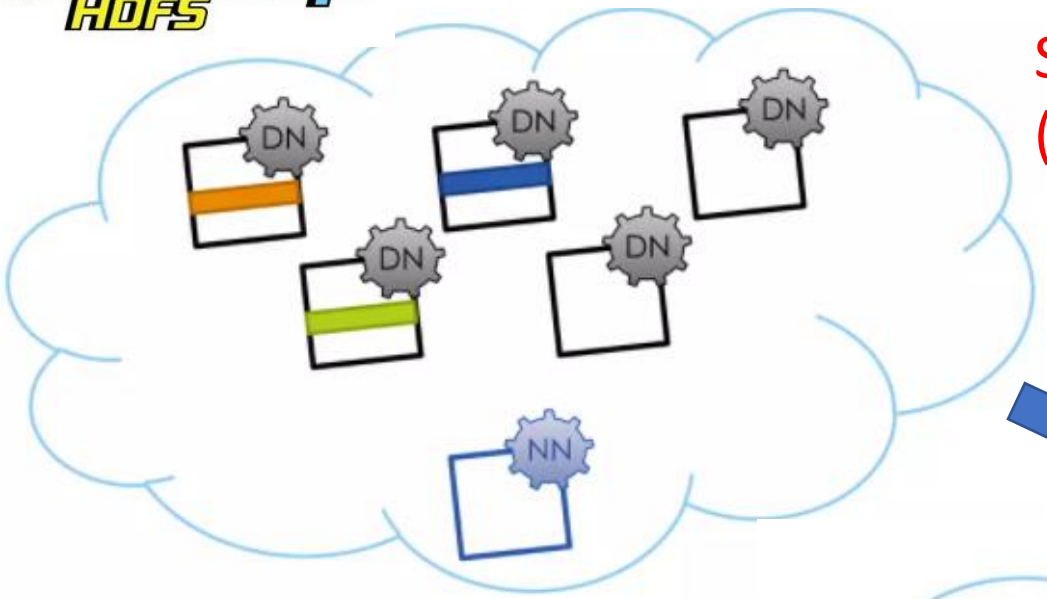


HDFS: Architecture



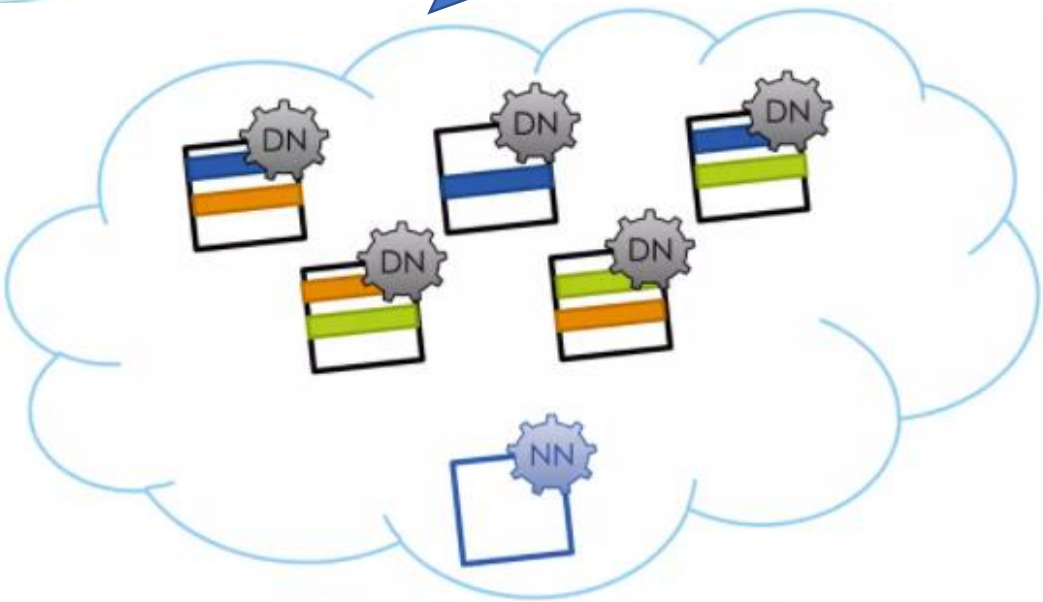
- Le nœud maître (**Name Node (NN)**) est l'orchestrateur, il contient et stocke les métadonnées de tout le cluster (leurs noms, blocs, localisation des données dans le cluster, etc)
- Le NN a une visibilité sur les nœuds esclaves.
- Le NN est le seul qui reçoit toutes les requêtes qui proviennent de client.

HDFS: Architecture



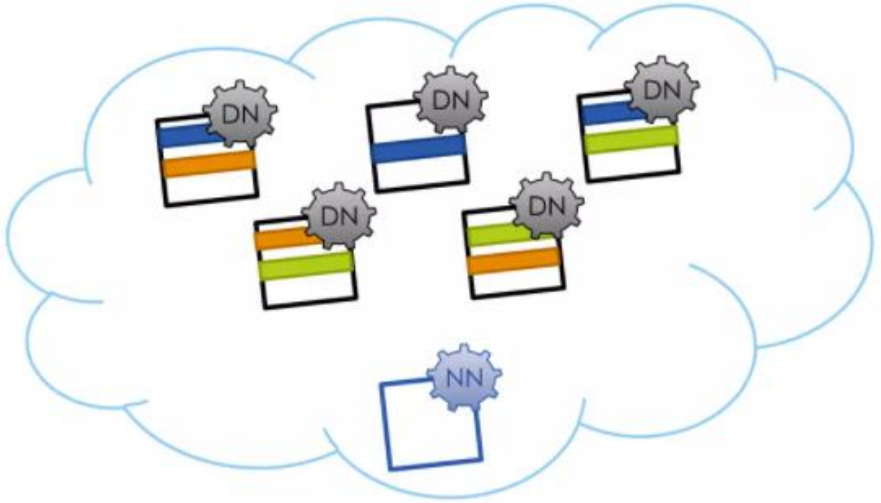
Si un des nœuds a un problème (tombe en panne)?

Réplication

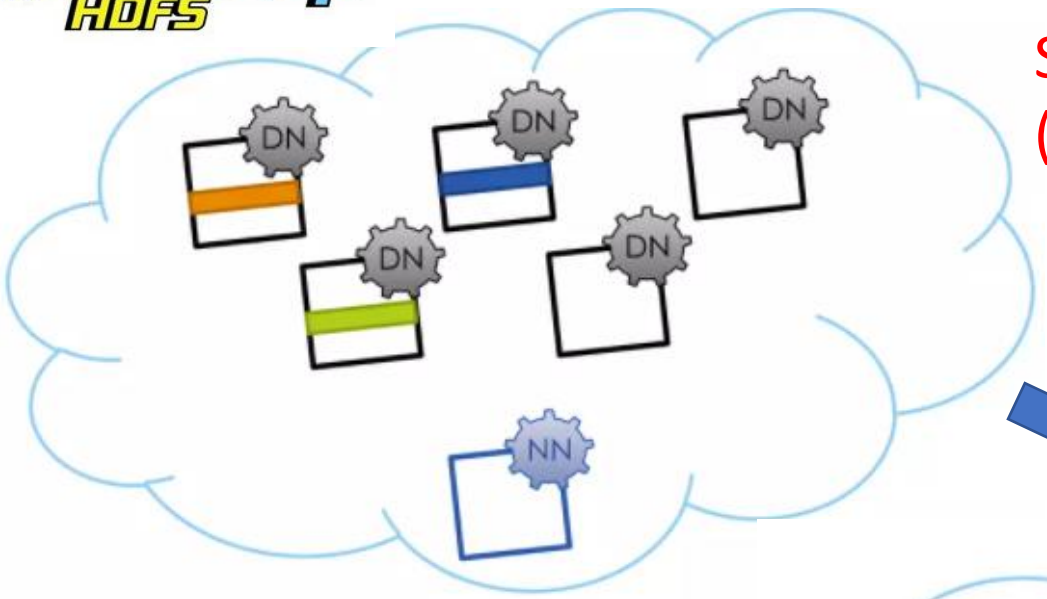


HDFS: Architecture

- Hadoop réplique chaque bloc 3 fois (Facteur de réplication =2)
- Il choisit 3 nœuds au hasard, et place une copie de bloc dans chacun d'eux
- Si le nœud est en panne, le NN détecte et s'occupe de répliquer encore les blocs qui y étaient hébergés pour avoir toujours 3 copies stockées.

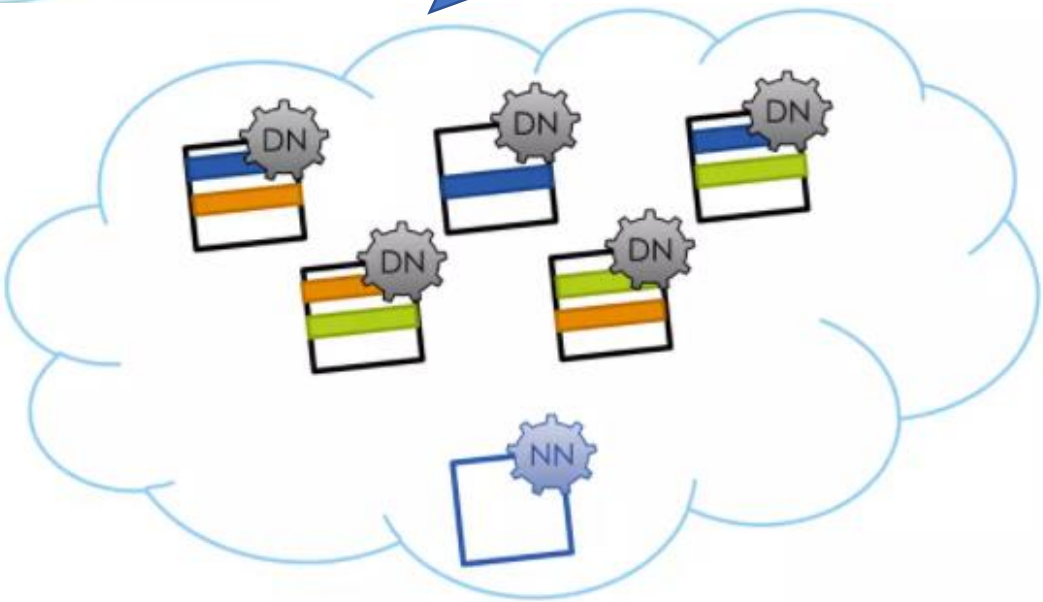


HDFS: Architecture

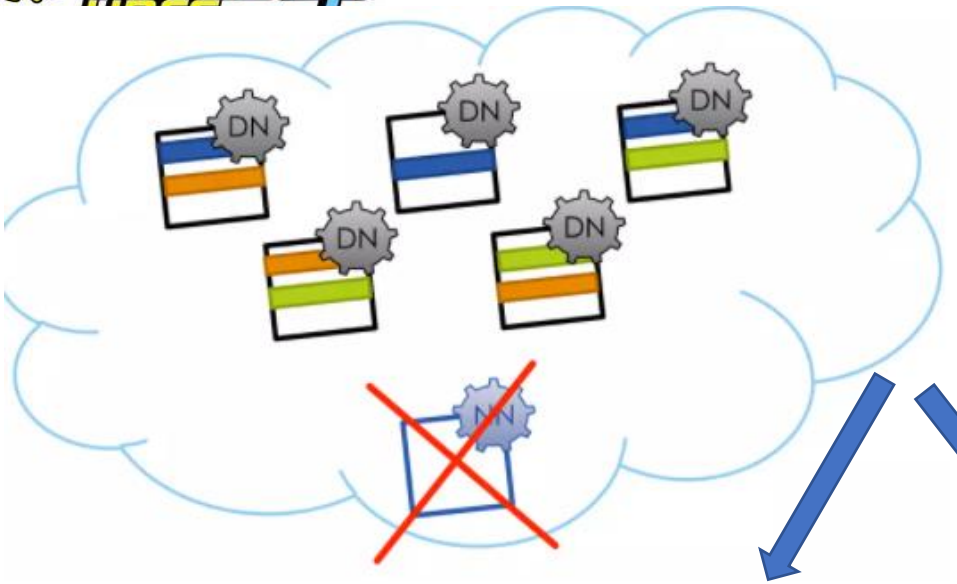


Si un des nœud a un problème (tombe en panne)?

Réplication



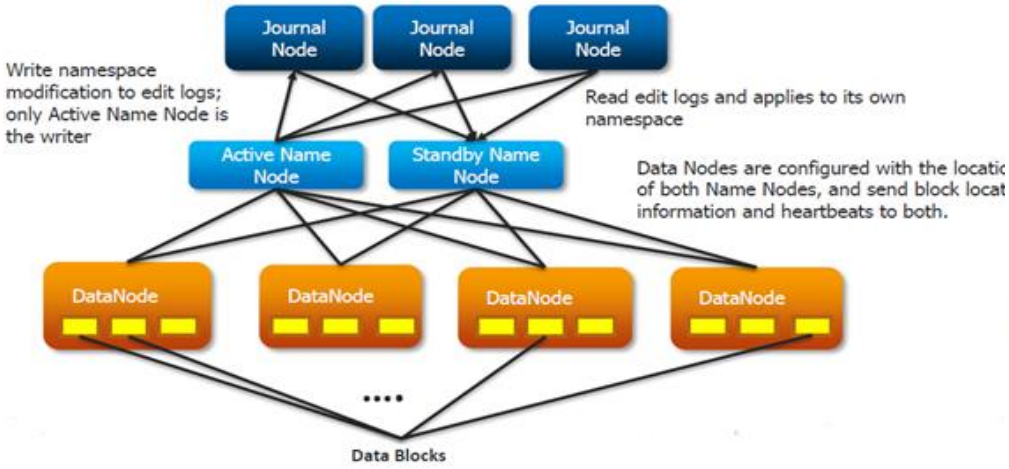
HDFS: Architecture



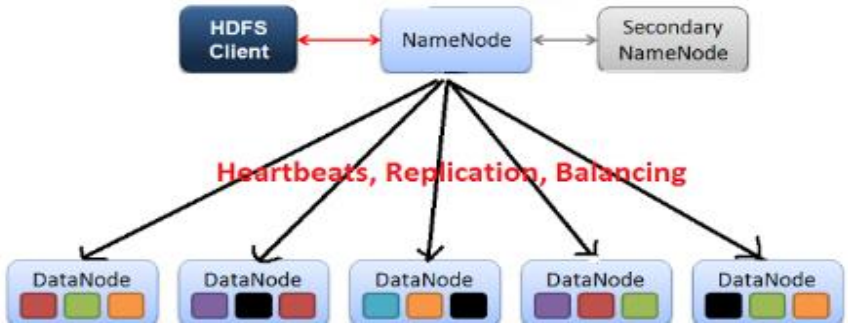
Si le Master tombe en panne?

+ NN (Plusieurs NN)

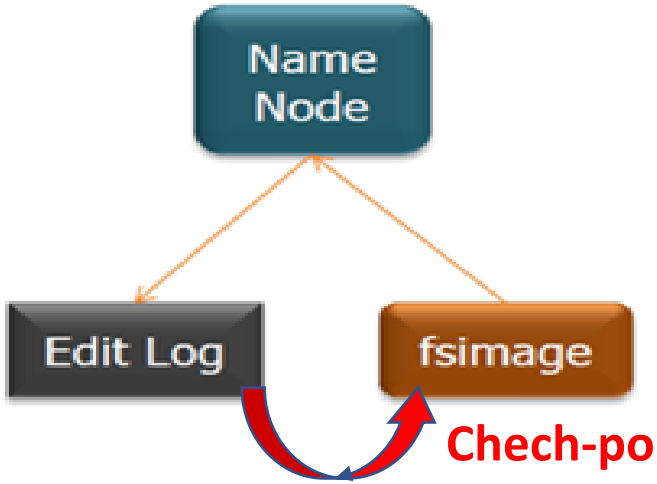
StdNN (StandBy Name Node)

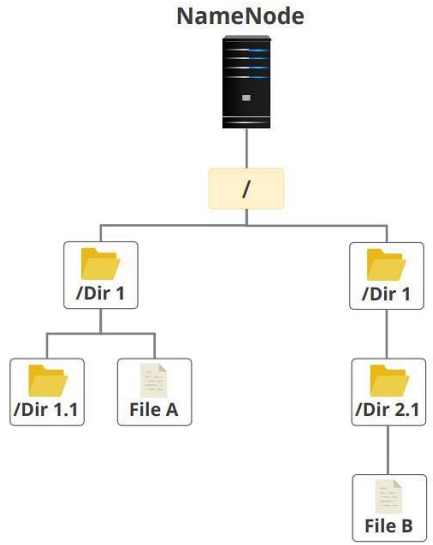


2NN (Secondary Name Node)



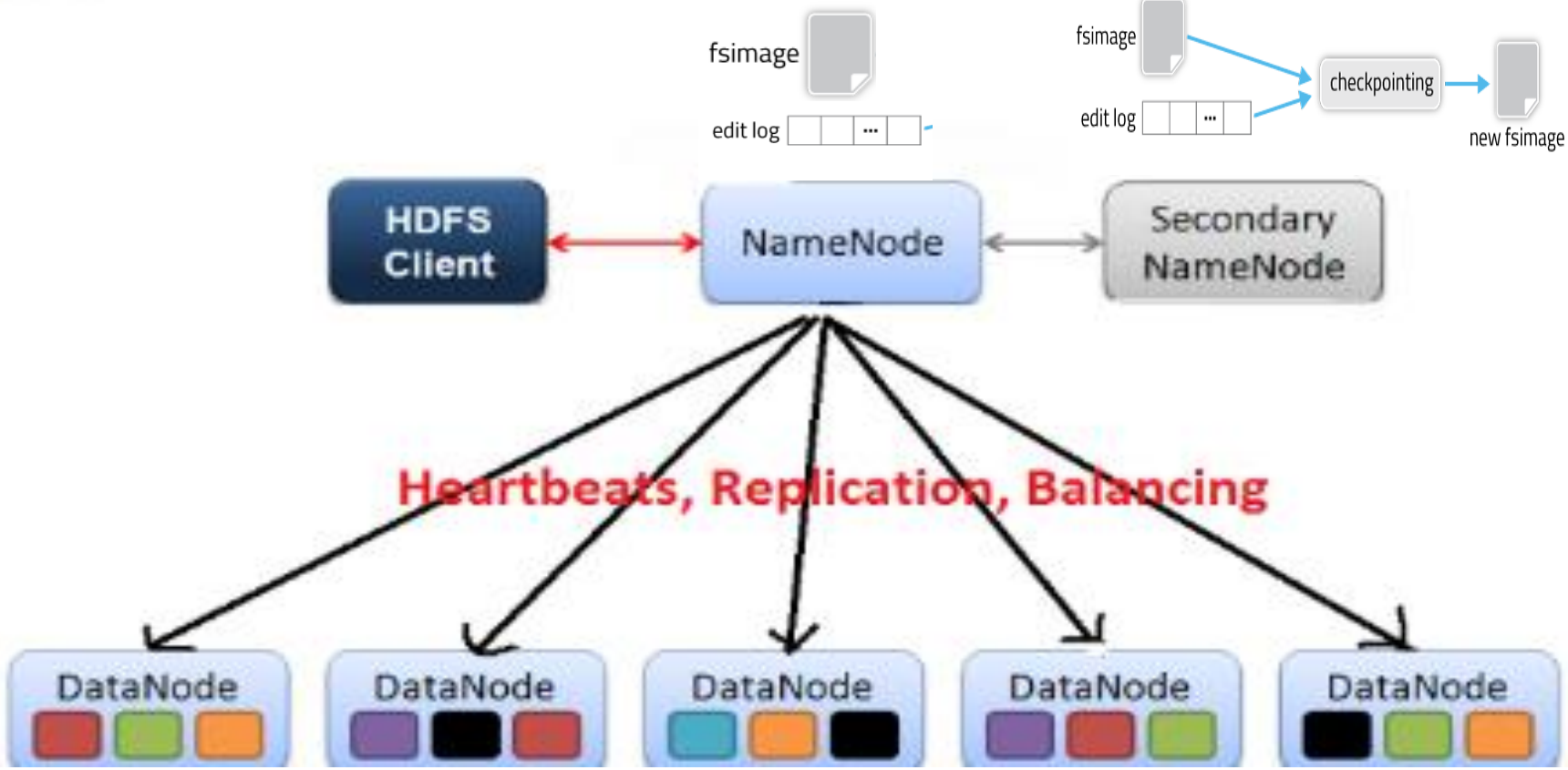
HDFS: Architecture





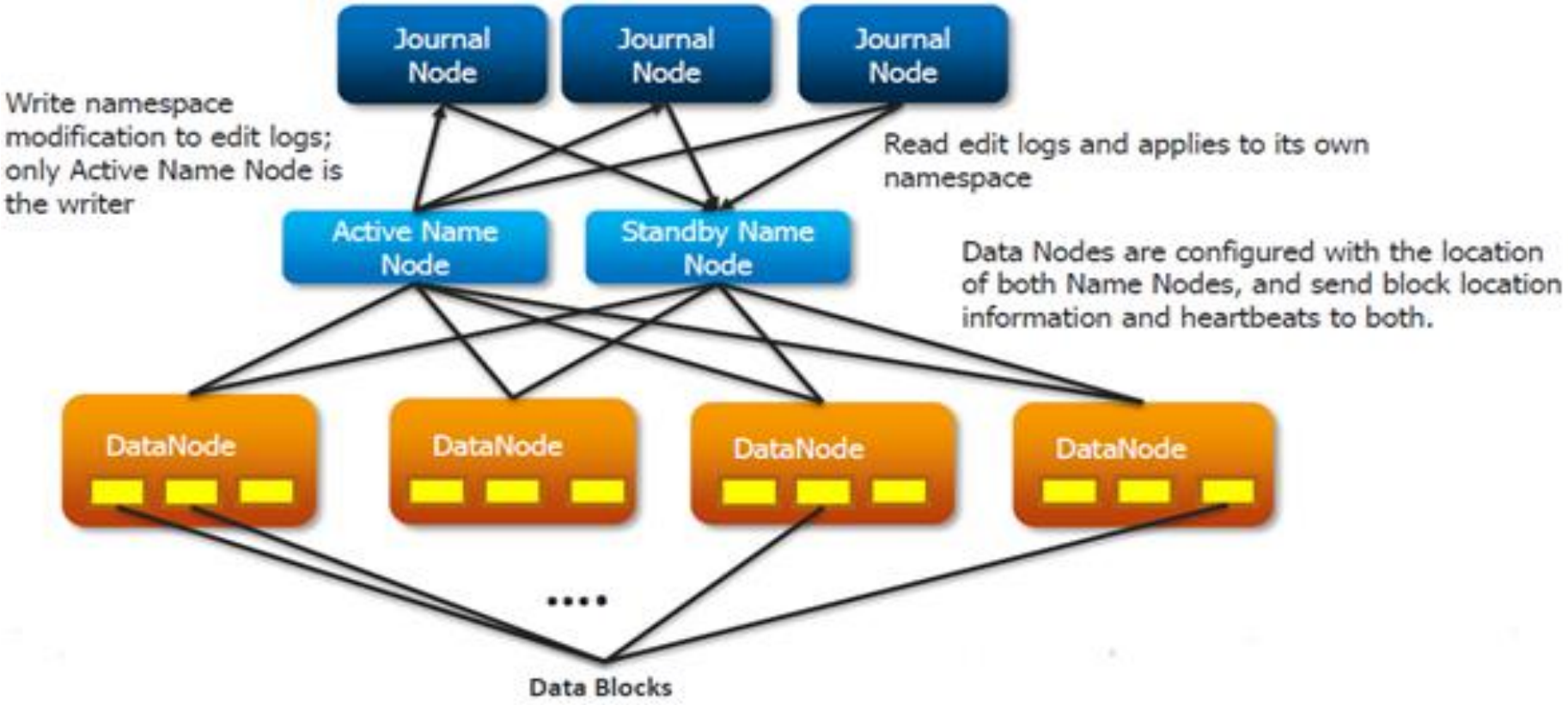
- Une **FsImage** (File system Image) est une image de l'état du système de fichiers à un moment donné de son fonctionnement. Le fichier FsImage, contient la structure de répertoires complète (espace de noms) du HDFS avec des informations telles que la liste des blocs stockés sur chaque nœud.
- Un **EditLog** ou journal de modifications est quant à lui un fichier dans lequel sont stockées les modifications apportées au système de fichiers HDFS ou toute action effectuée sur le cluster (création, réplication, suppression d'un bloc, etc.) depuis la création de la dernière FsImage.

HDFS: Architecture

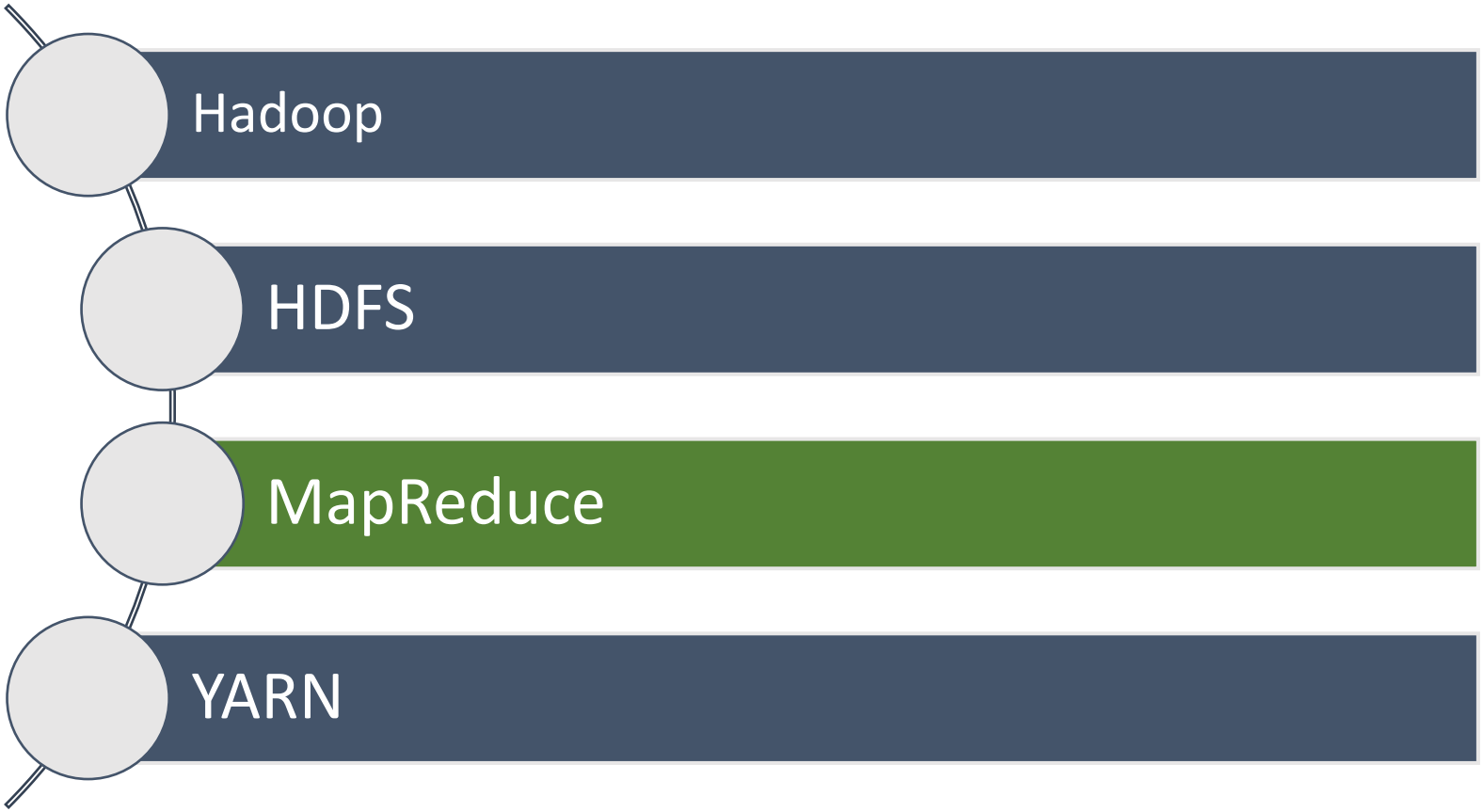


HDFS: Architecture

StdNN = 2NN + Distribution de charge → High availability



Plan



Map Reduce

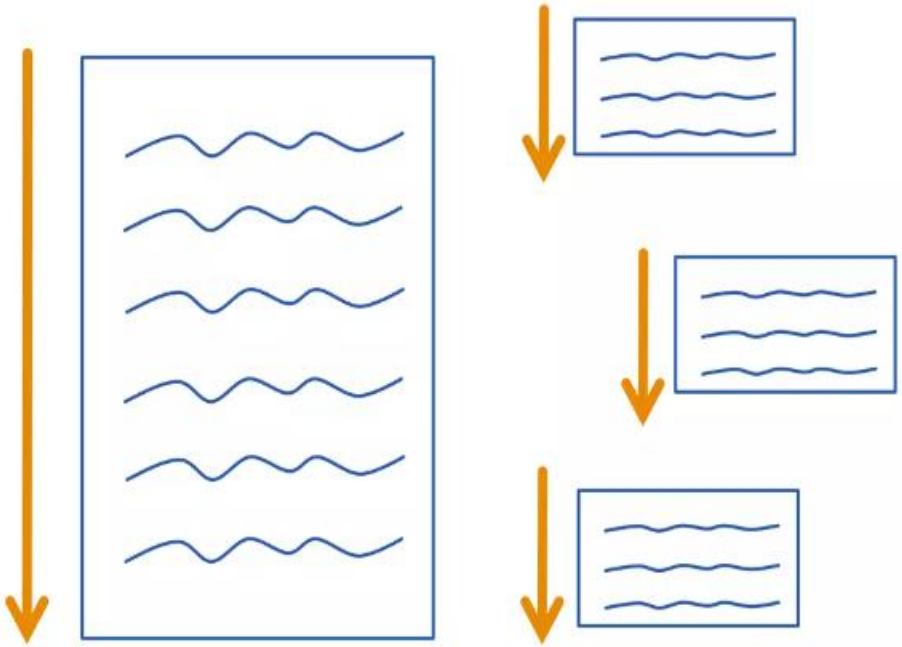
- **MapReduce** est un framework Java permettant d'écrire des programmes afin d'assurer le traitement parallèle des données massives.
- Un programme MapReduce comporte principalement deux tâches: map et reduce.

Map Reduce: Exemple Introductif

Patron d'architecture de développement permettant de traiter des données volumineuses de manière parallèle et distribuée

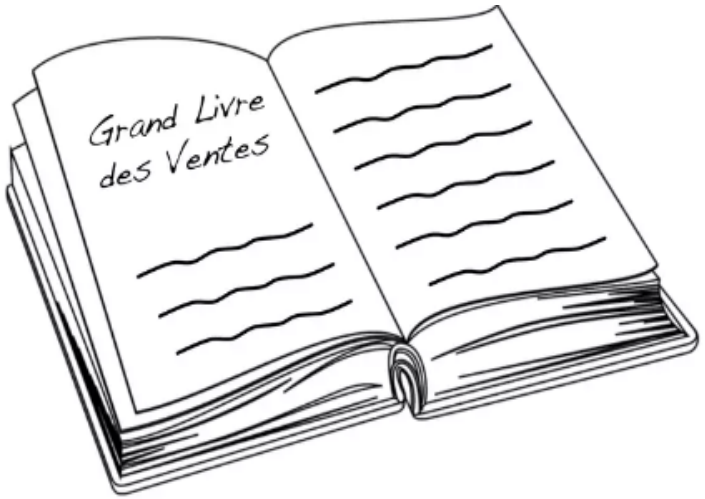
A la base, le langage Java est utilisé, mais grâce à une caractéristique de Hadoop appelée *Hadoop Streaming,* il est possible d'utiliser d'autres langages comme Python ou Ruby

Au lieu de parcourir le fichier séquentiellement (bcp de temps), il est divisé en morceaux qui sont parcourus en parallèle.



Map Reduce: Exemple Introductif

- Imaginons que vous ayez plusieurs magasins que vous gérez à travers le monde
- Un très grand livre de comptes contenant TOUTES les ventes
- **Objectif** : Calculer le total des ventes par magasin pour l'année en cours
- Supposons que les lignes du livres aient la forme suivante:
  - Jour Ville produit Prix



2024-01-01	London	Books	51,3
2024-01-01	Paris	Music	22,5
2024-01-02	Rome	Clothes	100,1
2024-01-02	London	Clothes	46

Map Reduce: Exemple Introductif

2024-01-01	London	Books	51,3
2024-01-01	Paris	Music	22,5
2024-01-02	Rome	Clothes	100,1
2024-01-02	London	Clothes	46

Objectif:

Calculer le total des ventes par magasin pour l'année en cours

Possibilité:

- Pour chaque entrée, saisir la ville et le prix de vente
- Si on trouve une entrée avec une ville déjà saisie, on les regroupe en faisant la somme des ventes

Dans un environnement traditionnel, on fera des Hashtable sous la forme <clé-valeur>. Dans ce cas, la clé sera l'adresse du magasin et la valeur le total de ventes

Map Reduce: Exemple Introductif

Objectif:

Calculer le total des ventes par magasin pour l'année en cours

2024-01-01	London	Books	51,3
2024-01-01	Paris	Music	22,5
2024-01-02	Rome	Clothes	100,1
2024-01-02	London	Clothes	46

Calcul séquentiel

London	51.3
Paris	22.5
Rome	100.1

London	97.3
Paris	22.5
Rome	100.1

Map Reduce: Exemple Introductif

Objectif:

Calculer le total des ventes par magasin pour l'année en cours

2024-01-01	London	Books	51,3
2024-01-01	Paris	Music	22,5
2024-01-02	Rome	Clothes	100,1
2024-01-02	London	Clothes	46

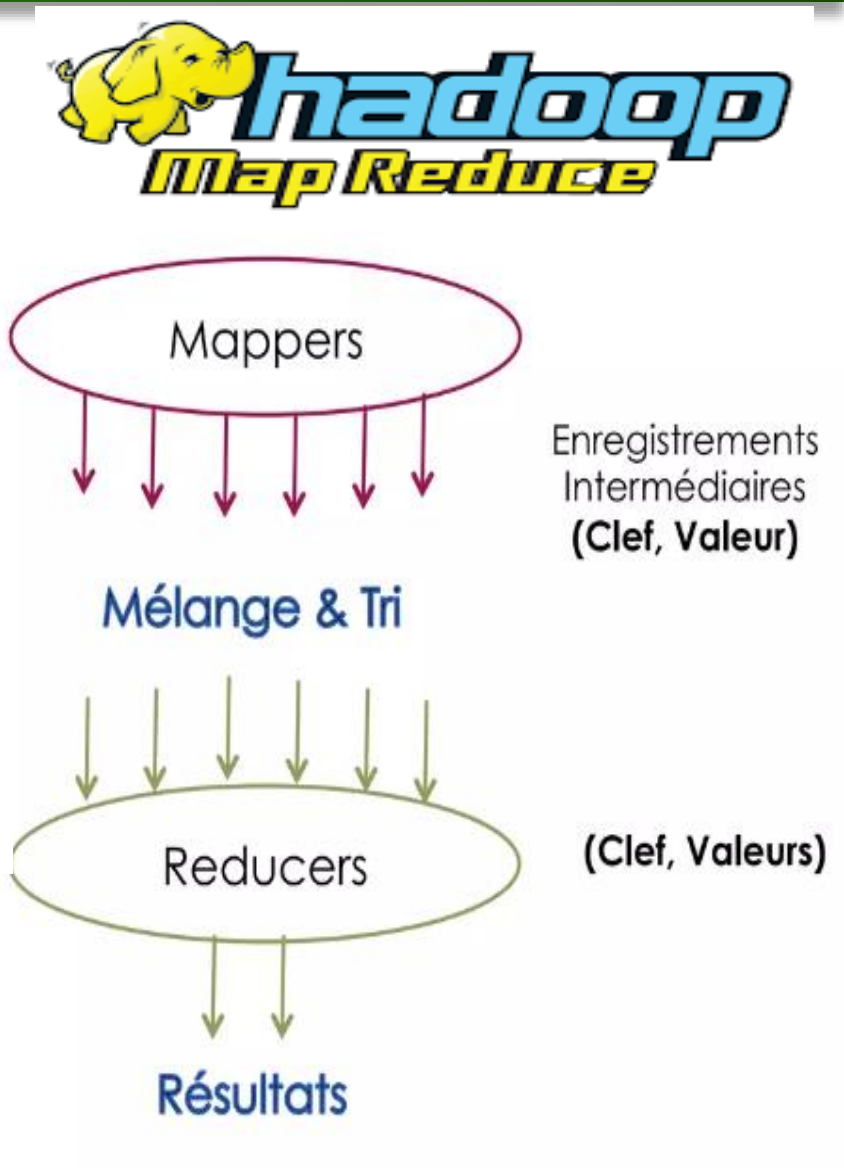
Étapes d'un job MapReduce:

- Les **mappers** sont de petits programmes qui commencent par traiter chacun une petite partie des données
- Ils fonctionnent en parallèle
- Leurs sorties représentent les enregistrements intermédiaires: sous forme d'un couple (clef, valeur)
- Une étape de Mélange et Tri s'ensuit

Mélange : Sélection des piles de fiches à partir des Mappers

Tri : Rangement des piles par ordre au niveau de chaque Reducer

- Chaque Reducer traite un ensemble d'enregistrements à la fois, pour générer les résultats finaux)



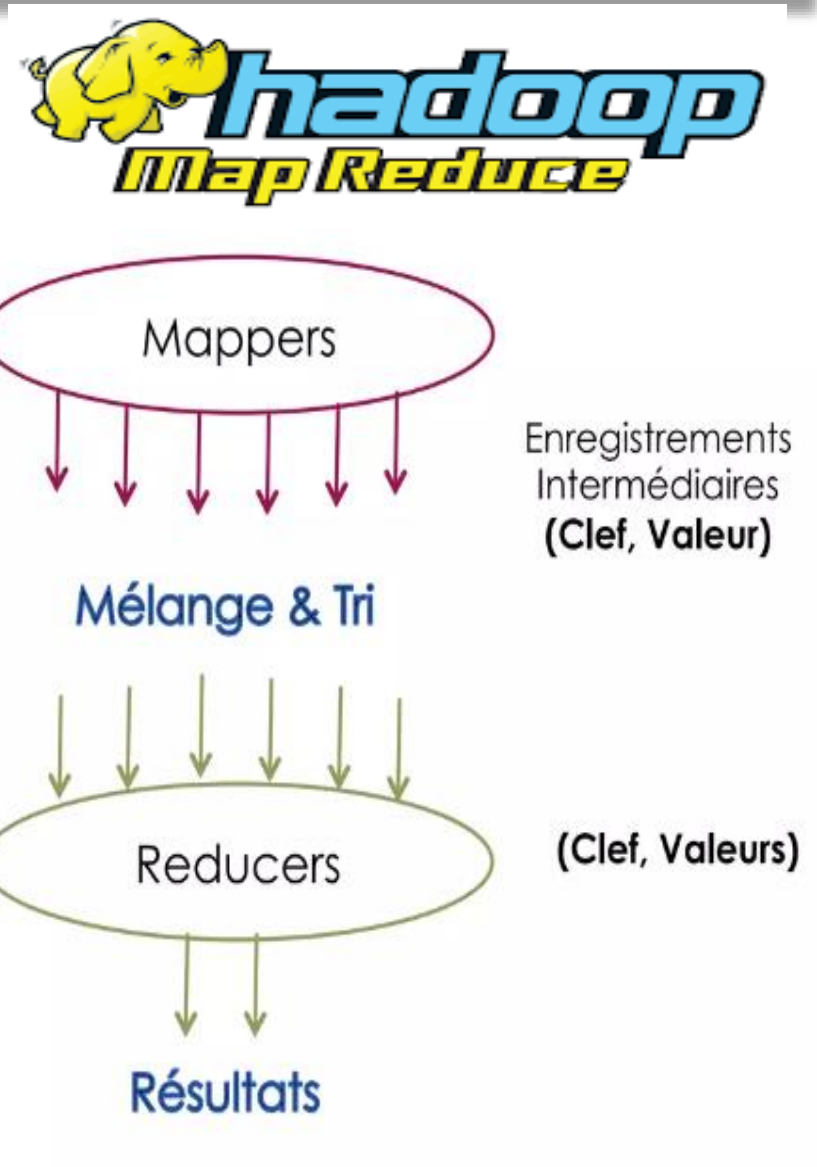
Map-Reduce: Fonctionnement

- Les **mappers** sont de petits programmes qui commencent par traiter chacun une petite partie des données
- Ils fonctionnent en parallèle
- Leurs sorties représentent les enregistrements intermédiaires: sous forme d'un couple (clef, valeur)
- Une étape de Mélange et Tri s'ensuit

Mélange : Sélection des piles de fiches à partir des Mappers

Tri : Rangement des piles par ordre au niveau de chaque Reducer

- Chaque Reducer traite un ensemble d'enregistrements à la fois, pour générer les résultats finaux)

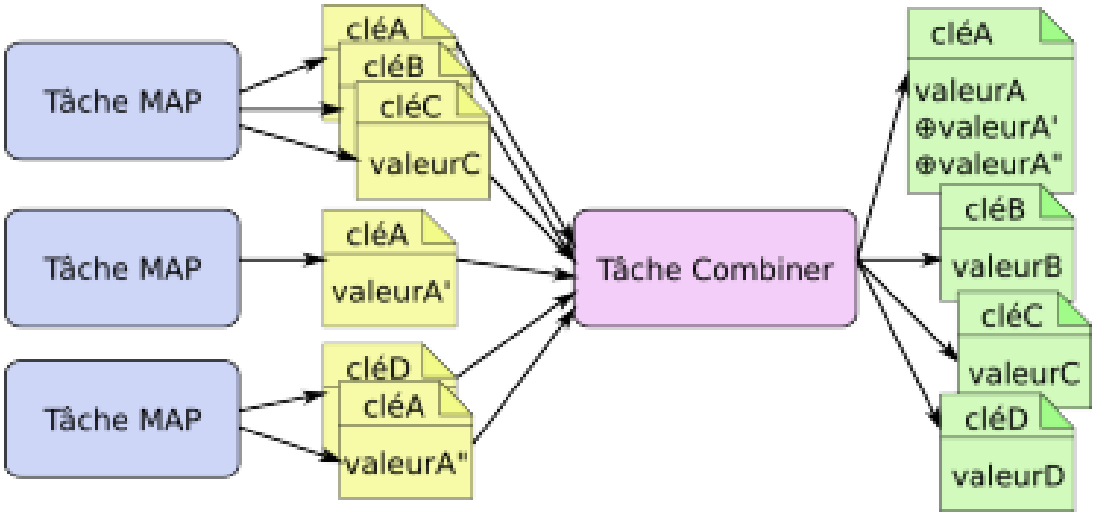


Combiner: entre Map et Reduce

Quand on traite des données volumineuses, ça peut devenir trop lent. Hadoop propose un troisième intervenant, entre map et reduce qui effectue un traitement local des paires produites par map. C'est le « **Combiner** ».

Son travail est de faire une première étape de réduction de tout ce qui est produit sur une machine.

Combiner traite des paires ayant la même clé sur la même machine que les tâches Map. Les paires qu'il émet sont envoyées aux reducers.

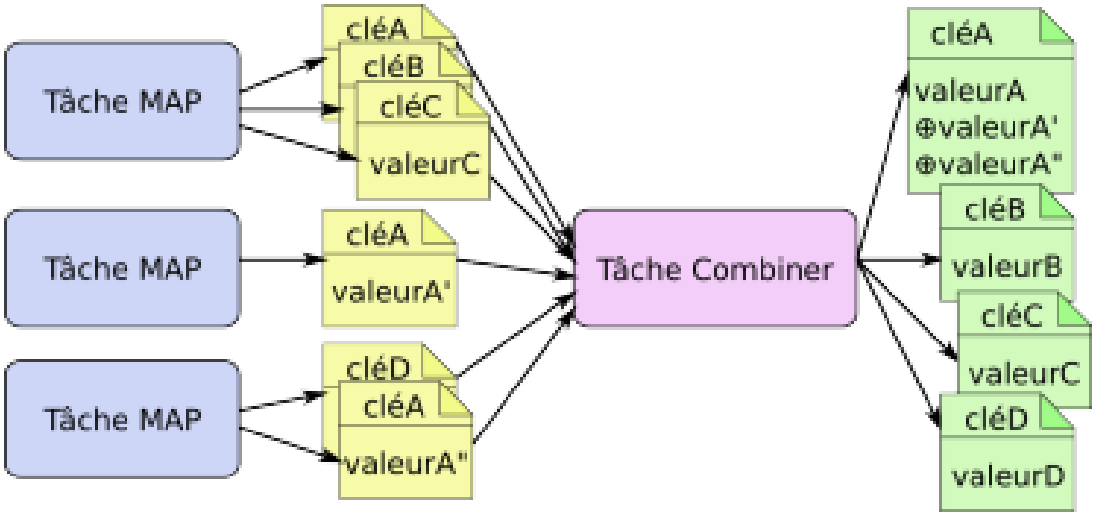


Combiner: entre Map et Reduce

Quand on traite des données volumineuses, ça peut devenir trop lent. Hadoop propose un troisième intervenant, entre map et reduce qui effectue un traitement local des paires produites par map. C'est le « **Combiner** ».

Son travail est de faire une première étape de réduction de tout ce qui est produit sur une machine.

Combiner traite des paires ayant la même clé sur la même machine que les tâches Map. Les paires qu'il émet sont envoyées aux reducers.



Combiner: entre Map et Reduce

Quand o … r trop lent.
Hadoop … reduce qui
effectue …
le « **Comb**
Son trava… t ce qui est
produit s…
Combiner… ine que les
tâches M…

Attention:

On ne peut pas employer de combiner quand l'opérateur d'agrégation n'est pas commutatif ou pas associatif.

Les opérateurs somme, min et max sont commutatifs et associatifs, mais pas le calcul d'une moyenne.

Combiner vs Reducer

- Les paramètres d'entrée et de sortie du Combiner doivent être identiques à ceux de sortie du Mapper, tandis que les types des paramètres de sortie du Reducer peuvent être différents de ceux de son entrée.
- On ne peut pas employer un Combiner quand la fonction n'est pas commutative et associative.
- Les Combiners reçoivent leurs paires d'un seul Mapper, tandis que les Reducers reçoivent les paires de tous les Combiners et/ou tous les Mappers. Les Combiners ont une vue restreinte des données

Map Reduce: Exemple 1 Word Count

Word Count !

Compter le nombre d'occurrences de chaque mot dans un ensemble de textes.

Données : un ensemble de textes

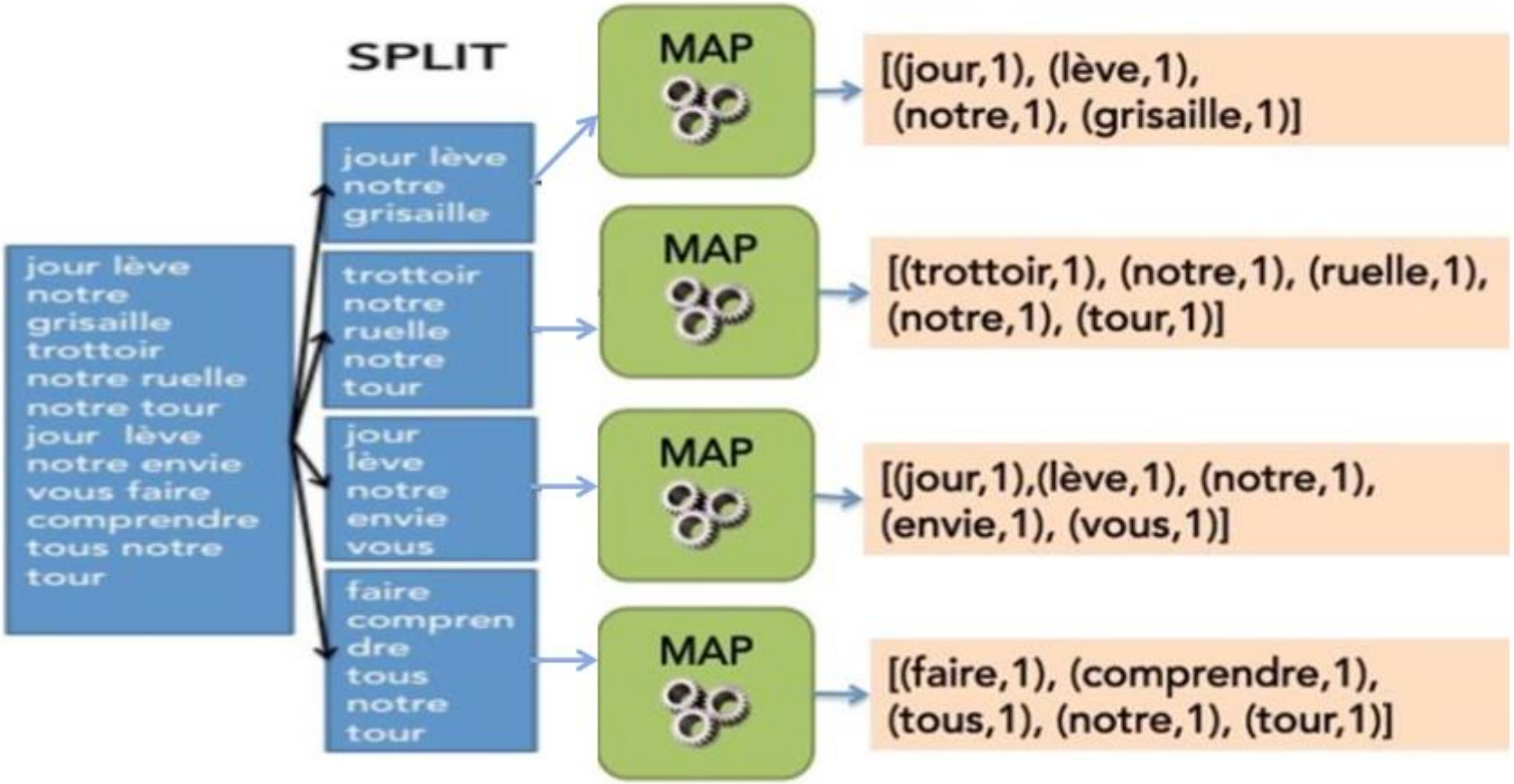
Le jour se lève sur notre grisaille, sur les trottoirs de nos ruelles et sur nos tours
[...]

Le jour se lève sur notre envie de vous faire comprendre à tous que c'est à notre tour
[...]

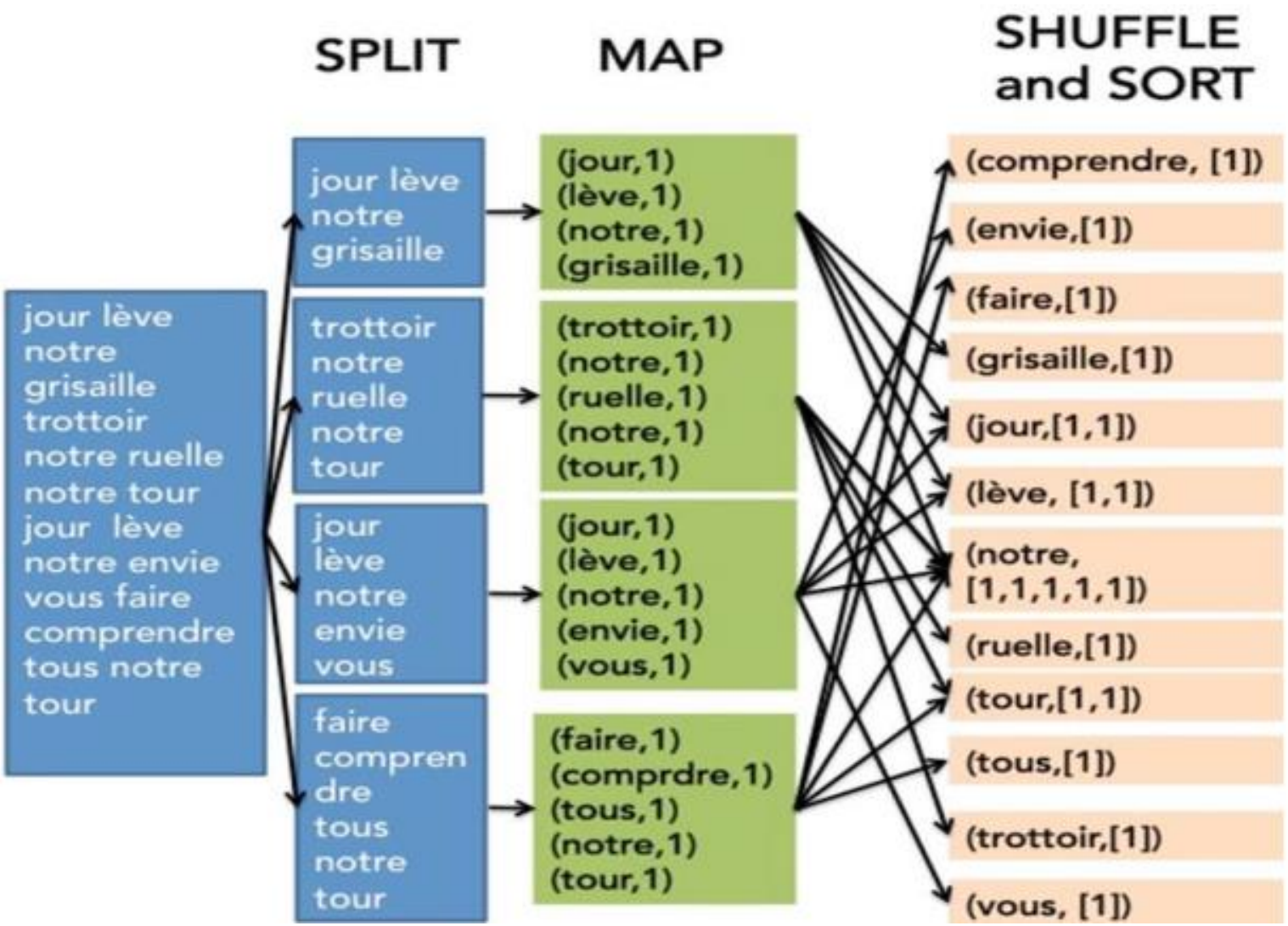
(**Grand Corps Malade,** Le Jour se lève. *Extrait*)

Map Reduce: Exemple 1 Word Count

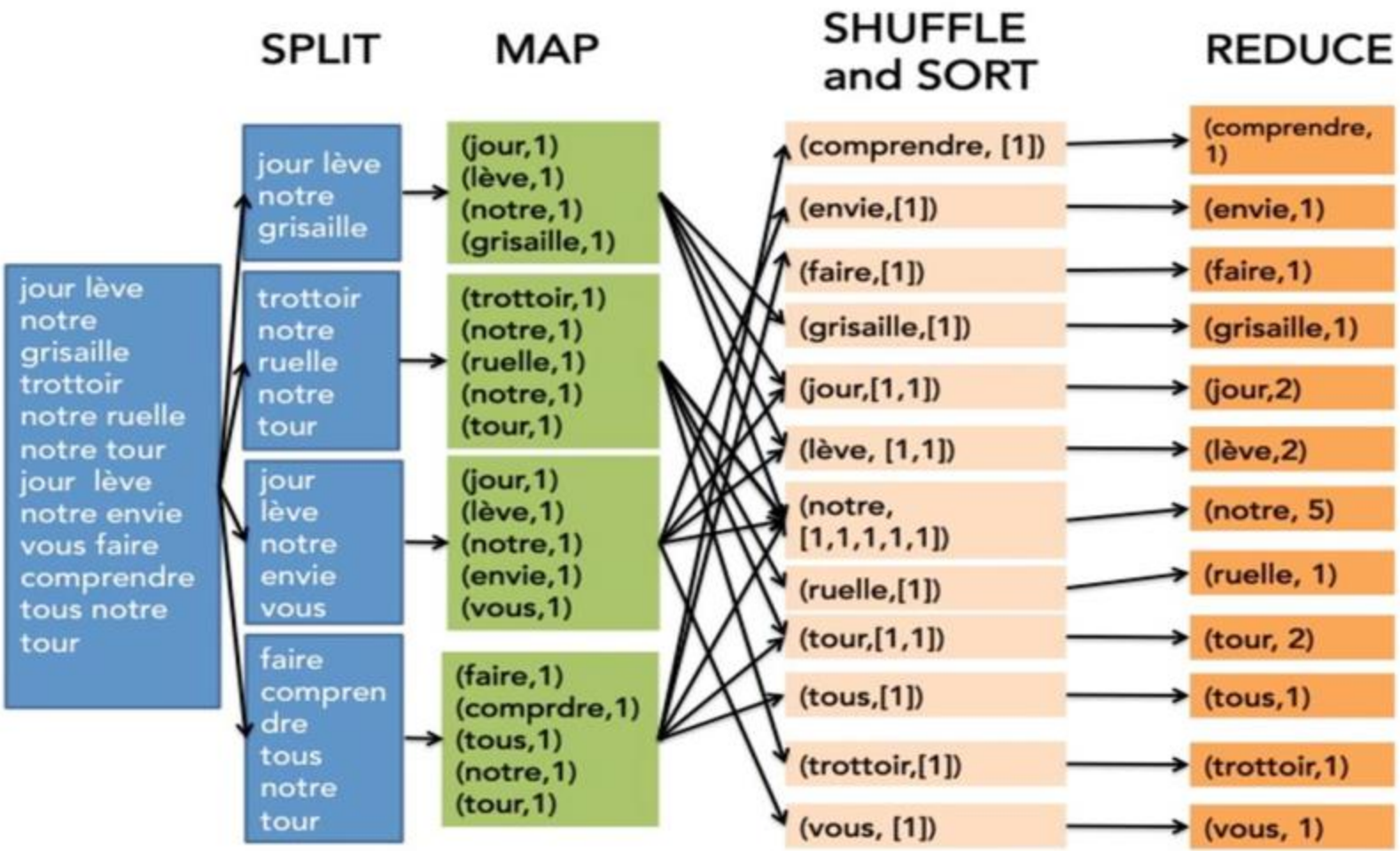
La fonction map()



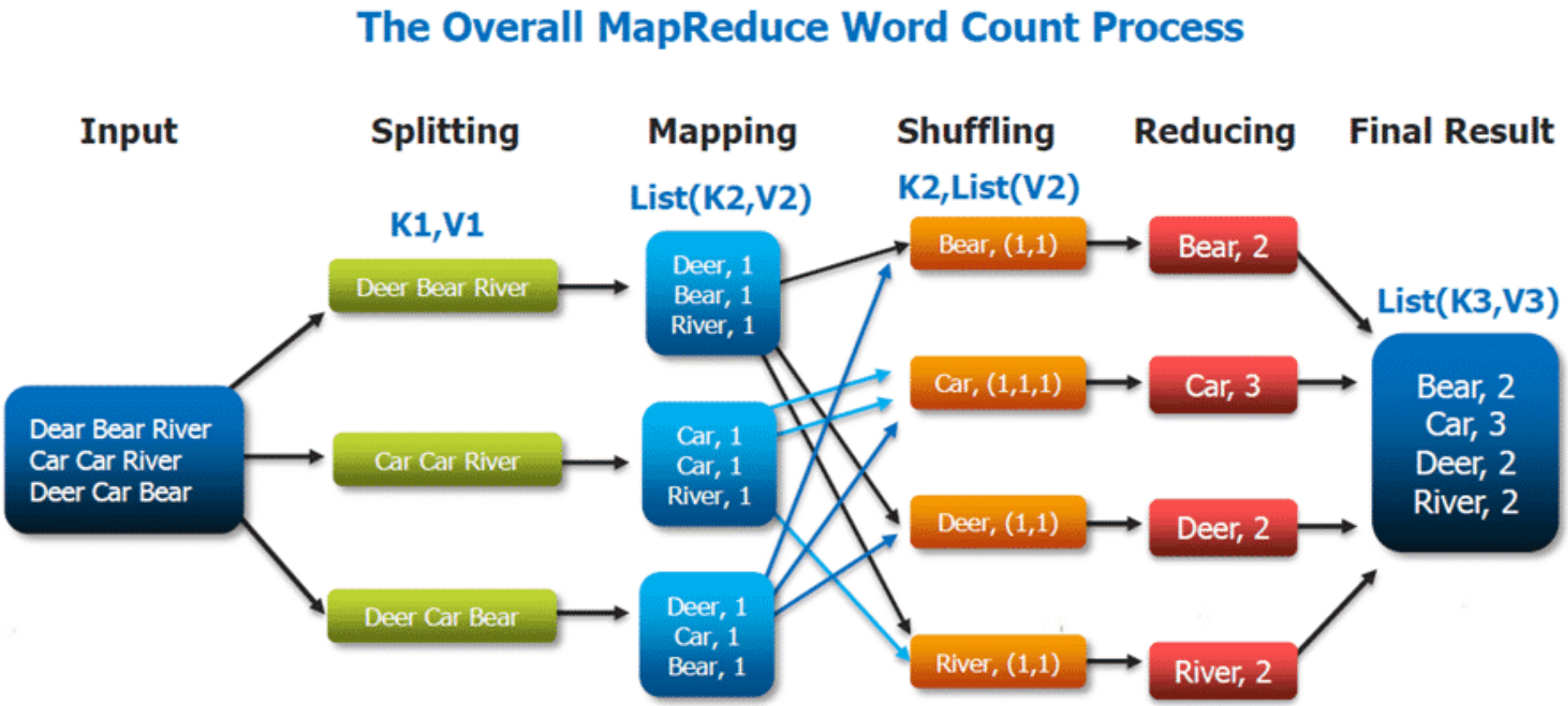
Map Reduce: Exemple 1 Word Count



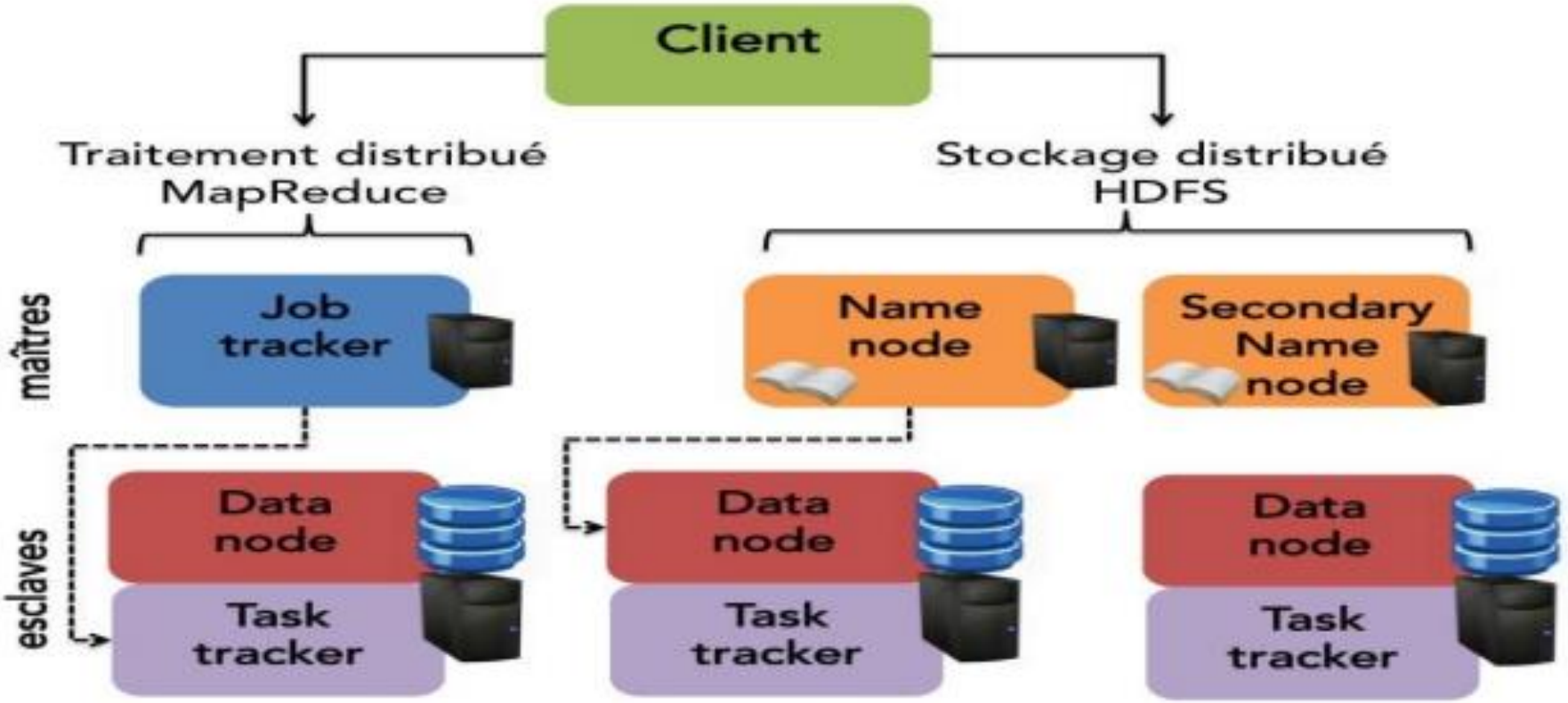
Map Reduce: Exemple 1 Word Count



Map Reduce: Exemple 2 Word Count



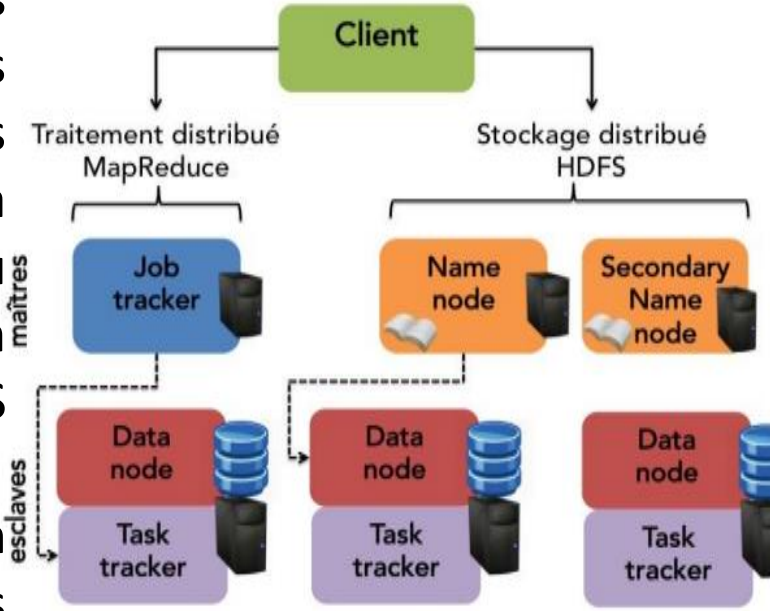
Hadoop et MapReduce



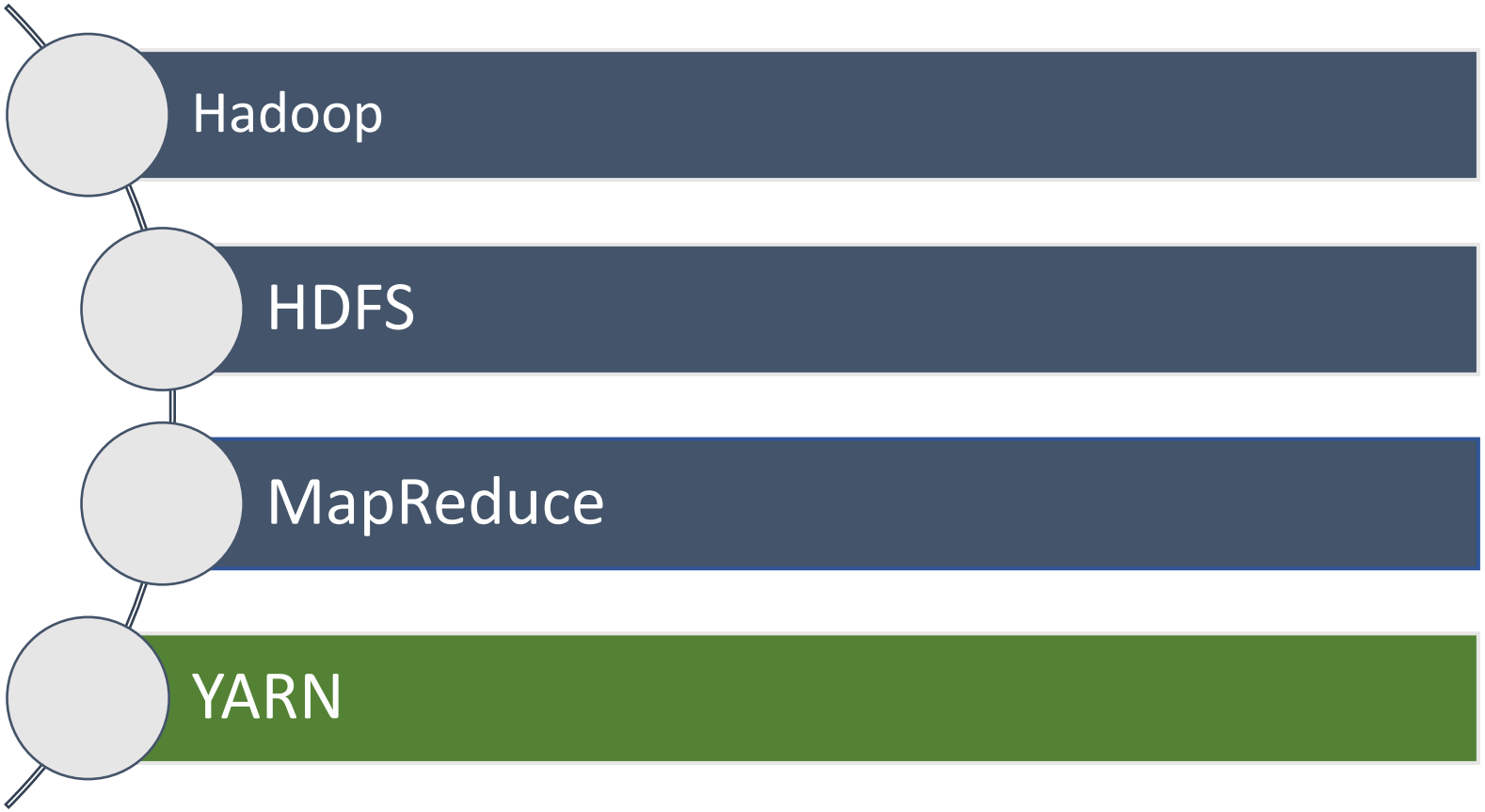
Hadoop et MapReduce

Comme pour HDFS, la gestion des tâches de Hadoop se base sur deux serveurs (des daemons):

- Le **JobTracker,** qui va directement recevoir la tâche à exécuter (un .jar Java), ainsi que les données d'entrées (nom des fichiers stockés sur HDFS) et le répertoire où stocker les données de sortie (toujours sur HDFS). Il y a un seul JobTracker sur une seule machine du cluster Hadoop. Le JobTracker est en communication avec le NameNode de HDFS et sait donc où sont les données.
- Le **TaskTracker**, qui est en communication constante avec le JobTracker et va recevoir les opérations simples à effectuer (MAP/REDUCE) ainsi que les blocs de données correspondants (stockés sur HDFS). Il y a un TaskTracker sur chaque machine du cluster.



Plan

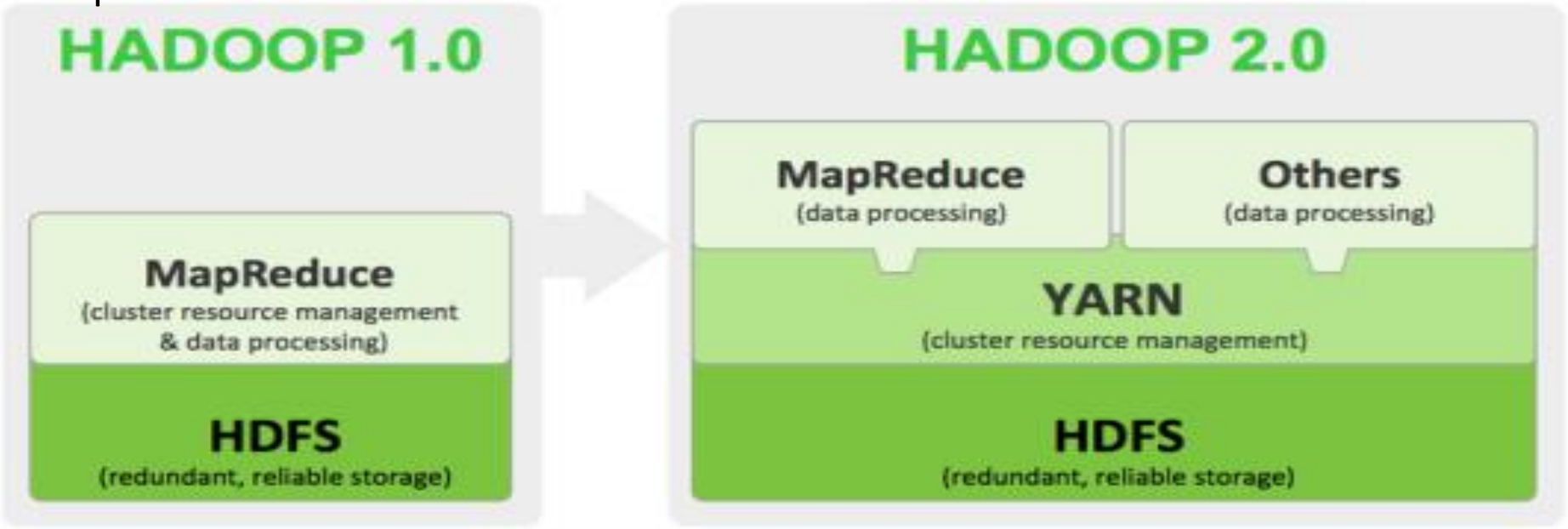


Hadoop , Map Reduce , YARN

La description de Hadoop comme possédant 2 couches (MapReduce et HDFS) est correcte pour la version 1 de Hadoop. Depuis la version 2, Hadoop a adopté une troisième couche : YARN ("Yet Another Resource Negociator"), un outil de gestion des ressources distribuée.

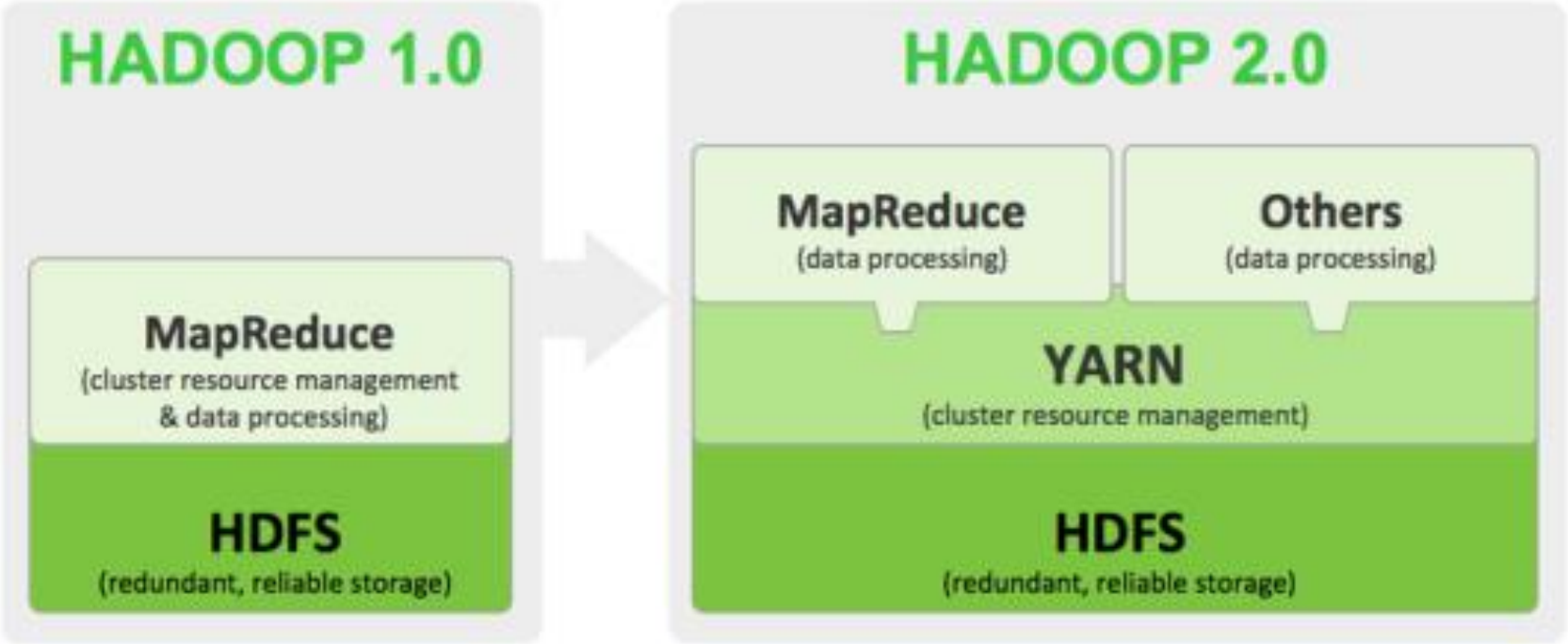
YARN provient d'un découpage de la première version de Hadoop MapReduce en deux sous-couches :

- 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

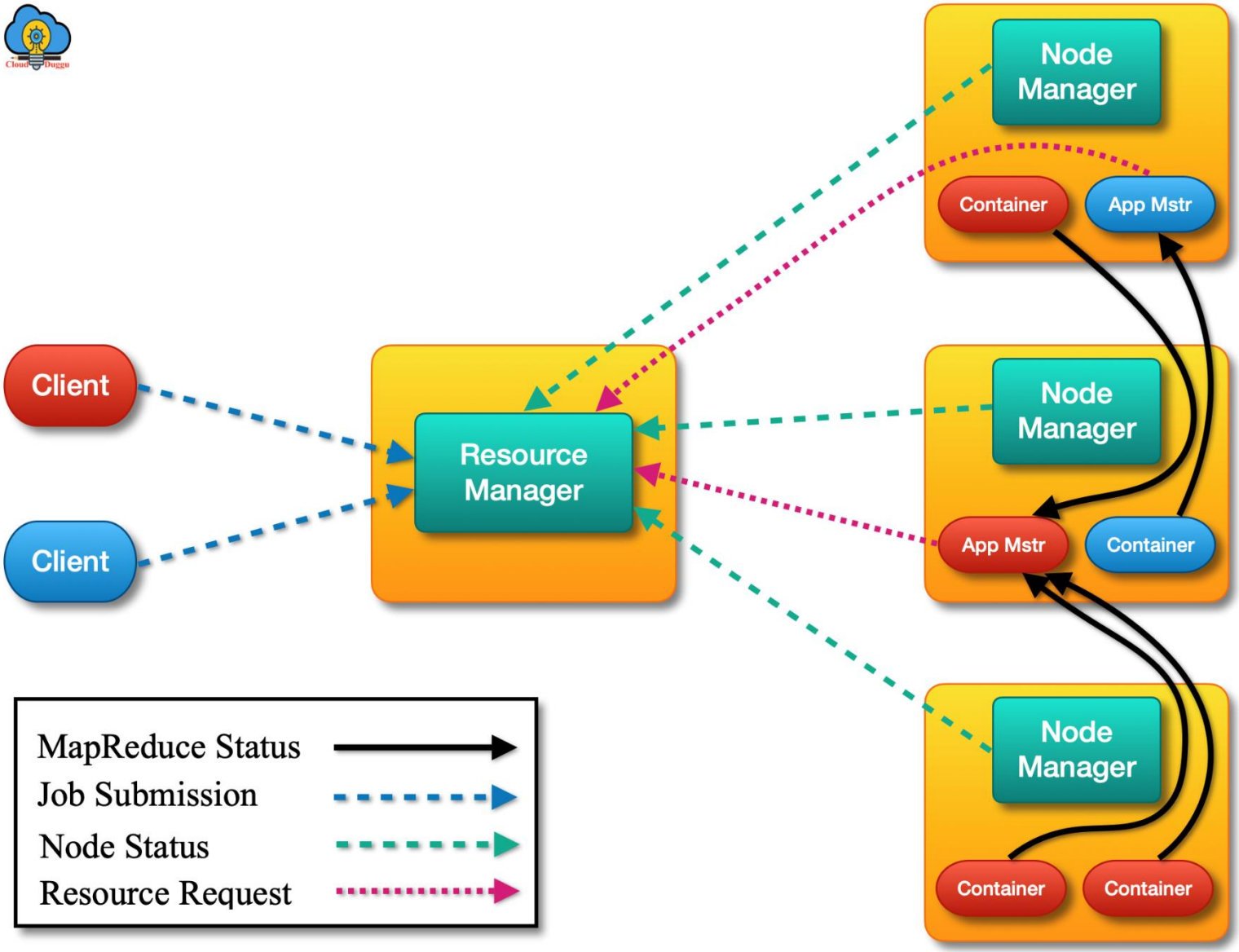


Hadoop , Map Reduce , YARN

Ce découpage a amené de nombreux autres outils (liés ou indépendants d'Apache Hadoop) à profiter de l'environnement HDFS comme moyen de stocker aisément de grandes quantités de données sans nécessairement MapReduce. Un écosystème d'outils liés à Hadoop a alors émergé et est de nos jours très développé.



Hadoop , Map Reduce , YARN



Références

- Rudi Bruchez. (2015), « Les bases de données NoSQL et le Big Data », Editeur : Eyrolles, ISBN : 978-2-212-14155-9
- Rudi Bruchez (2021), « Les bases de données NoSQL », Editeur : Eyrolles ISBN : 978-2-212-67866-6
- Juvénal Chokogoue. (2017). « Hadoop - Devenez opérationnel dans le monde du Big Data », Editeur: ENI, ISBN: 978-2409007613