Université Constantine 2
جامعة قسنطينة 2

TDSAI

Chapitre 3 : Hadoop
-Partie 01-

Dr. S.ZERABI
Faculté des NTIC
Soumeya.zerabi@univ-constantine2.dz

Etudiants concernés

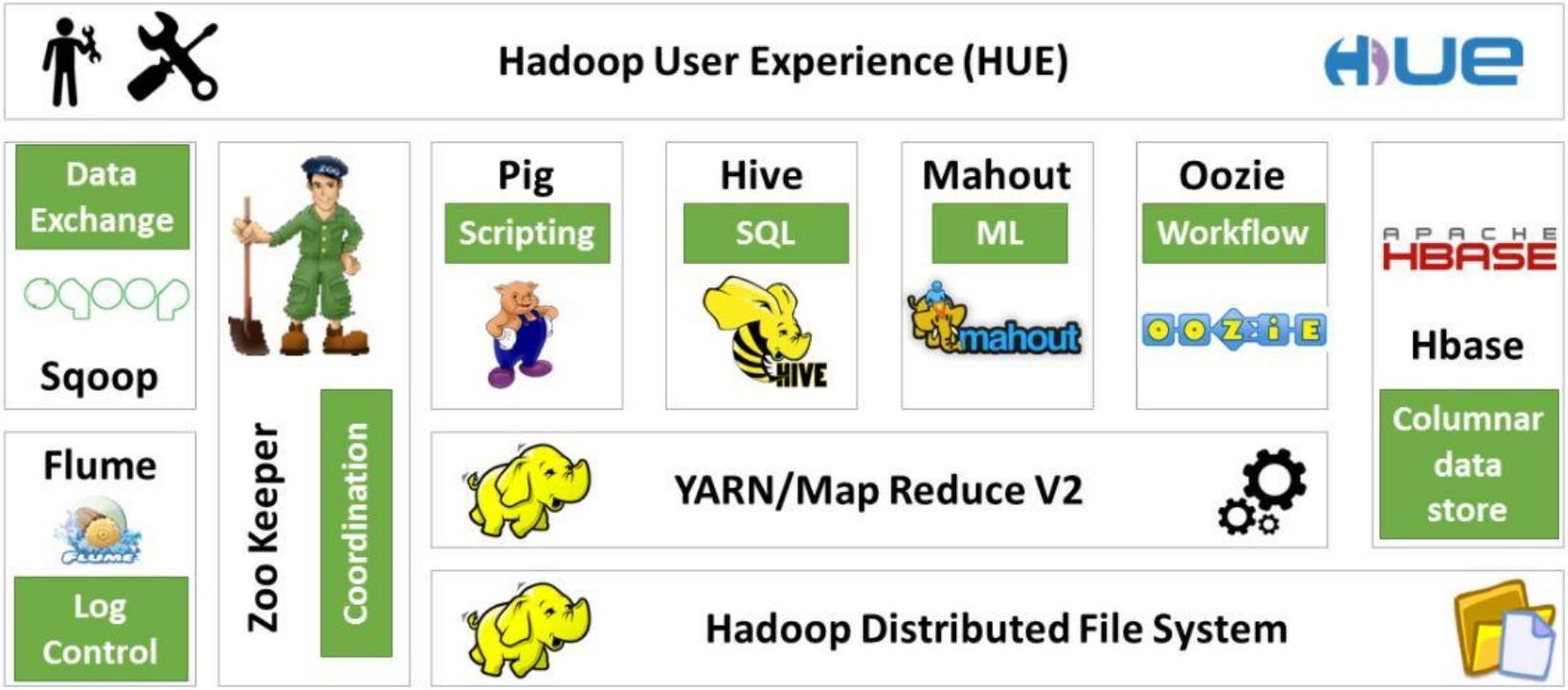
Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	M1	SDIA

Hadoop

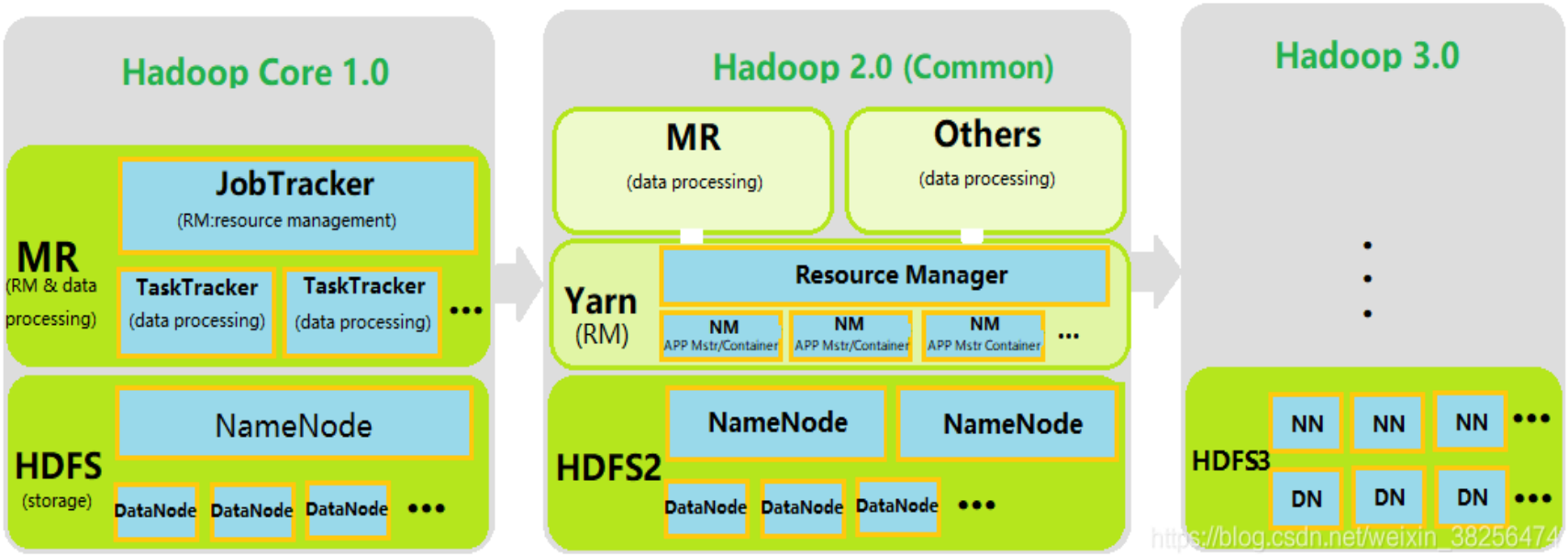
- **Hadoop** est un framework **Java** libre,
- destiné à faciliter la création d'applications **distribuées**, **parallèles**.
- permettant aux applications de travailler avec des milliers de nœuds et des pétaoctets de données.
- Hadoop a été inspiré des travaux de Google (GFS en 2003), MapReduce (2004), BigTable (2006).
- Il existe plusieurs sources open source et payantes de Hadoop:
  - Apache Hadoop (open source)
  - Cloudera (CDH)
  - Hortonworks
  - MapR
  - AWS
  - …..

Ecosystème de Hadoop

Hadoop comporte plusieurs frameworks



Hadoop

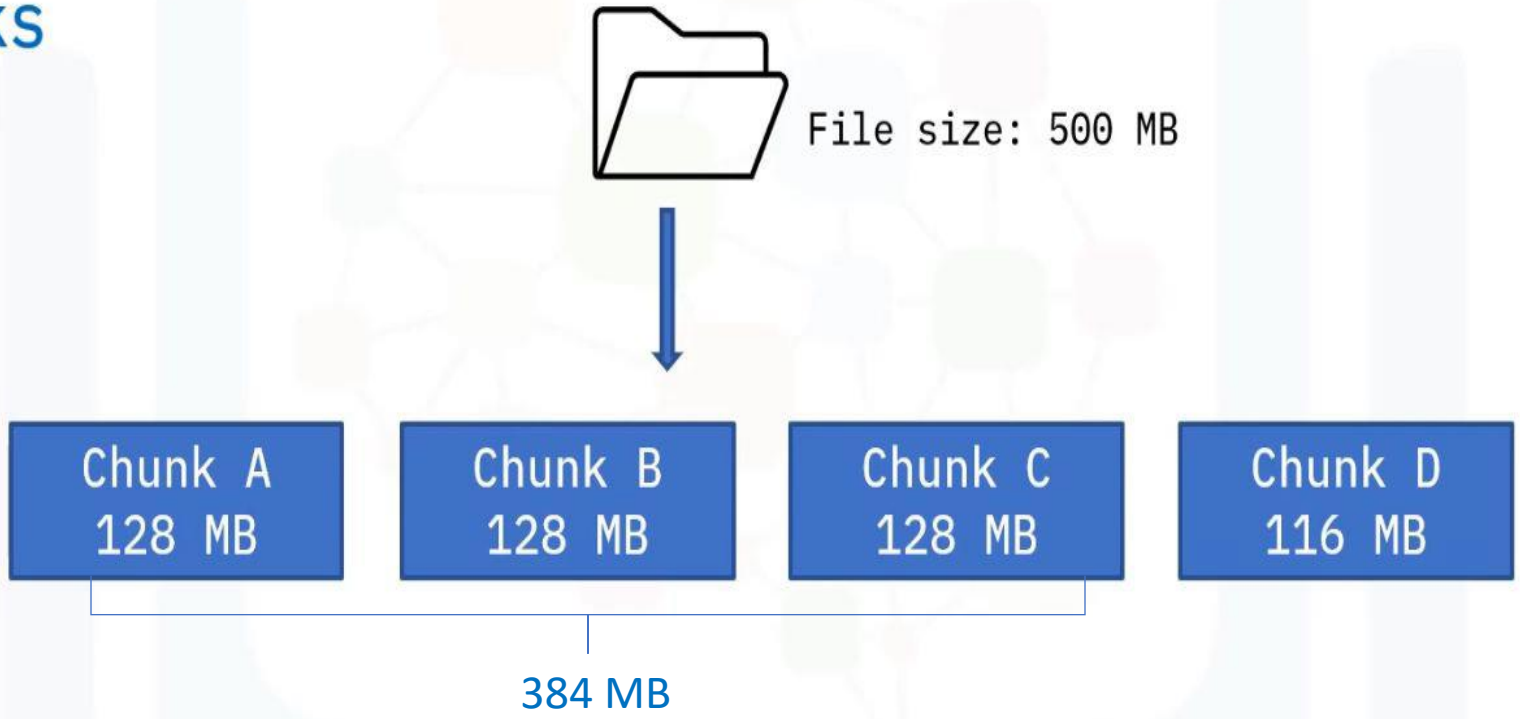


https://blog.csdn.net/weixin_38256474

HDFS

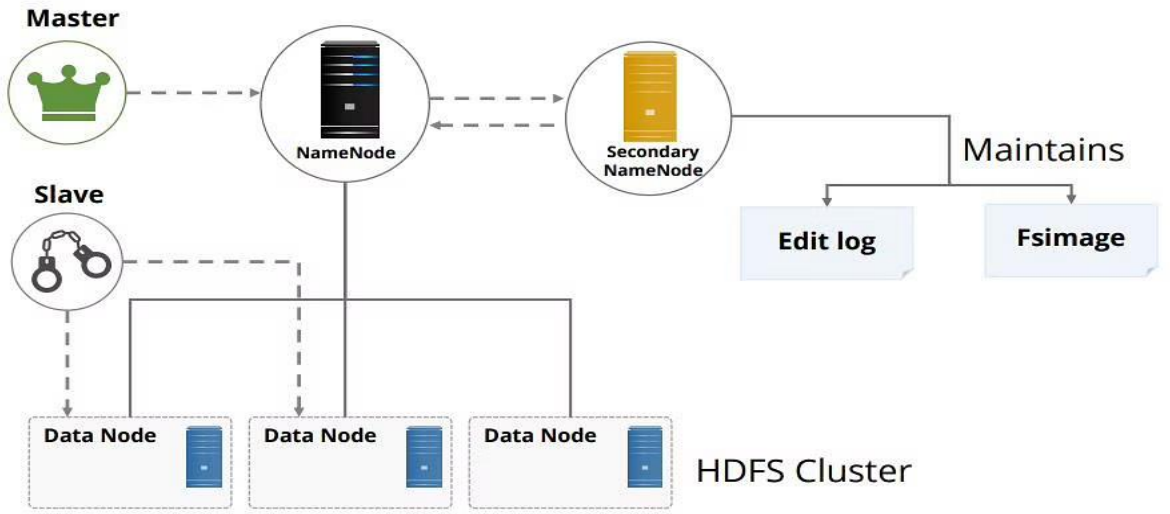
- Hadoop Distributed File System
- Il permet de stocker des fichiers très volumineux de manière **distribuée**.
- les fichiers sont physiquement découpés en **blocs** d'octets de grande taille (par défaut 128 Mo) pour optimiser les temps de transfert et d'accès ;
- Ces blocs sont ensuite **répartis**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

Blocks


File size: 500 MB
Chunk A
128 MB
Chunk B
128 MB
Chunk C
128 MB
Chunk D
116 MB
384 MB

HDFS

- Il suit une architecture *maître/esclave*.
- Le nœud maître (**NameNode**) est l'orchestrateur, il contient et stocke les métadonnées (leurs noms, blocs, localisation dans le cluster) des fichiers **« un gros annuaire ».**
- Le **secondary name node** sert de namenode de secours (en cas de panne du nœud maître). Il fait des sauvegardes régulières de l'annuaire.
- Les nœuds esclaves (**DataNode**) permettent la gestion des opérations de stockage locales (création, suppression et réplication de blocs) sous la supervision du namenode.



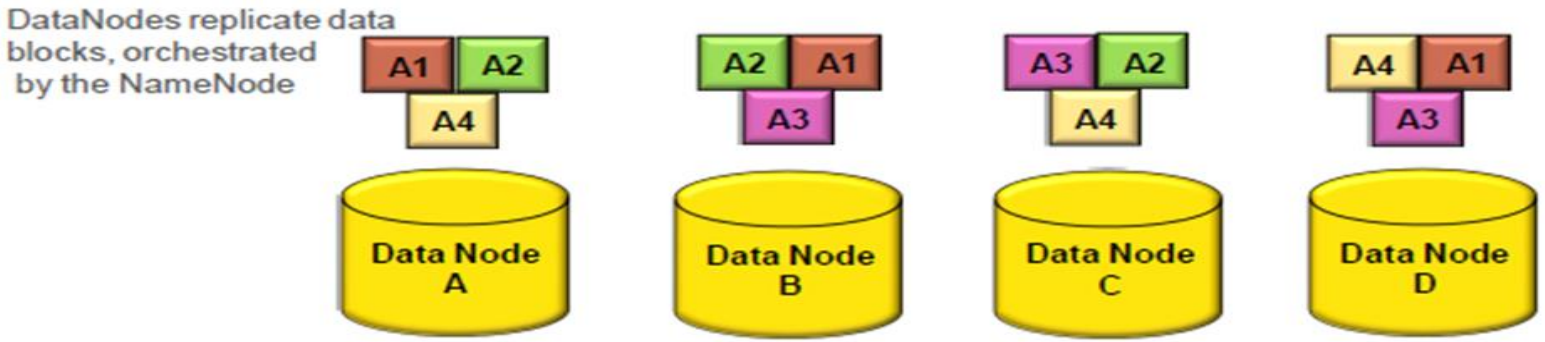
Caractéristiques de HDFS

- **Distribution**
- **Grand volume de données**
- **Tolérance aux fautes**
- **Haute disponibilité des données**
- **Evolutivité**

© Dr Zerabi S.

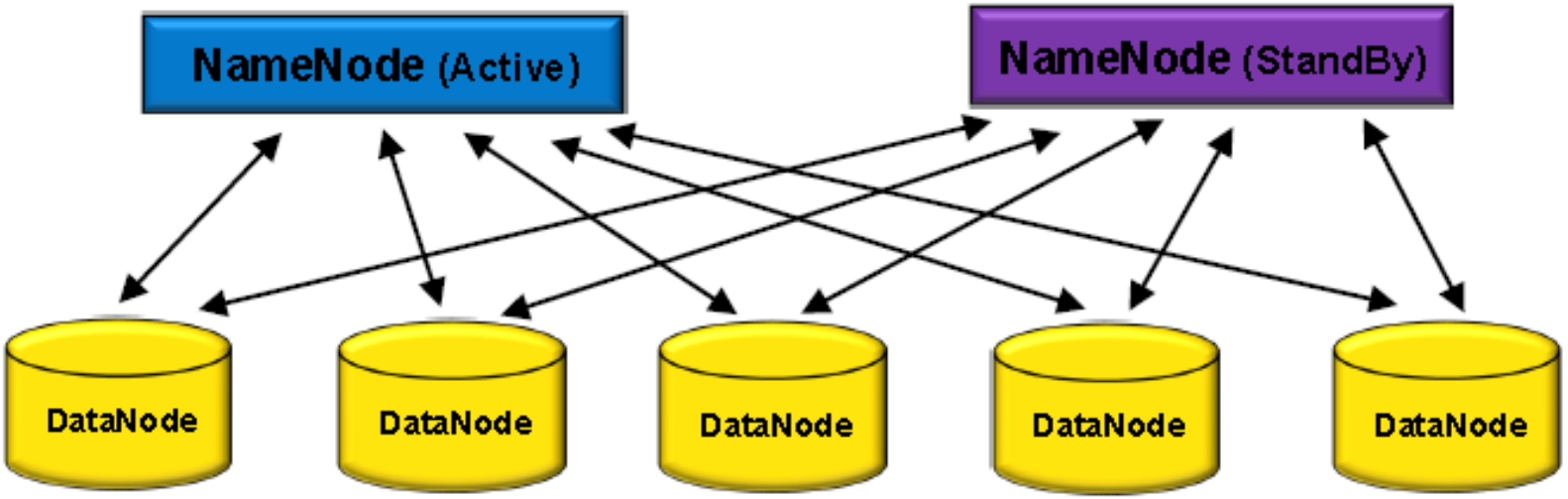
HDFS

- HDFS assure la **tolérance aux fautes.**
- La défaillance d'un nœud ne provoque pas l'échec de calcul.
- la réplication (de manière intelligente) de données.
- Par défaut le nombre de réplication est **3**.



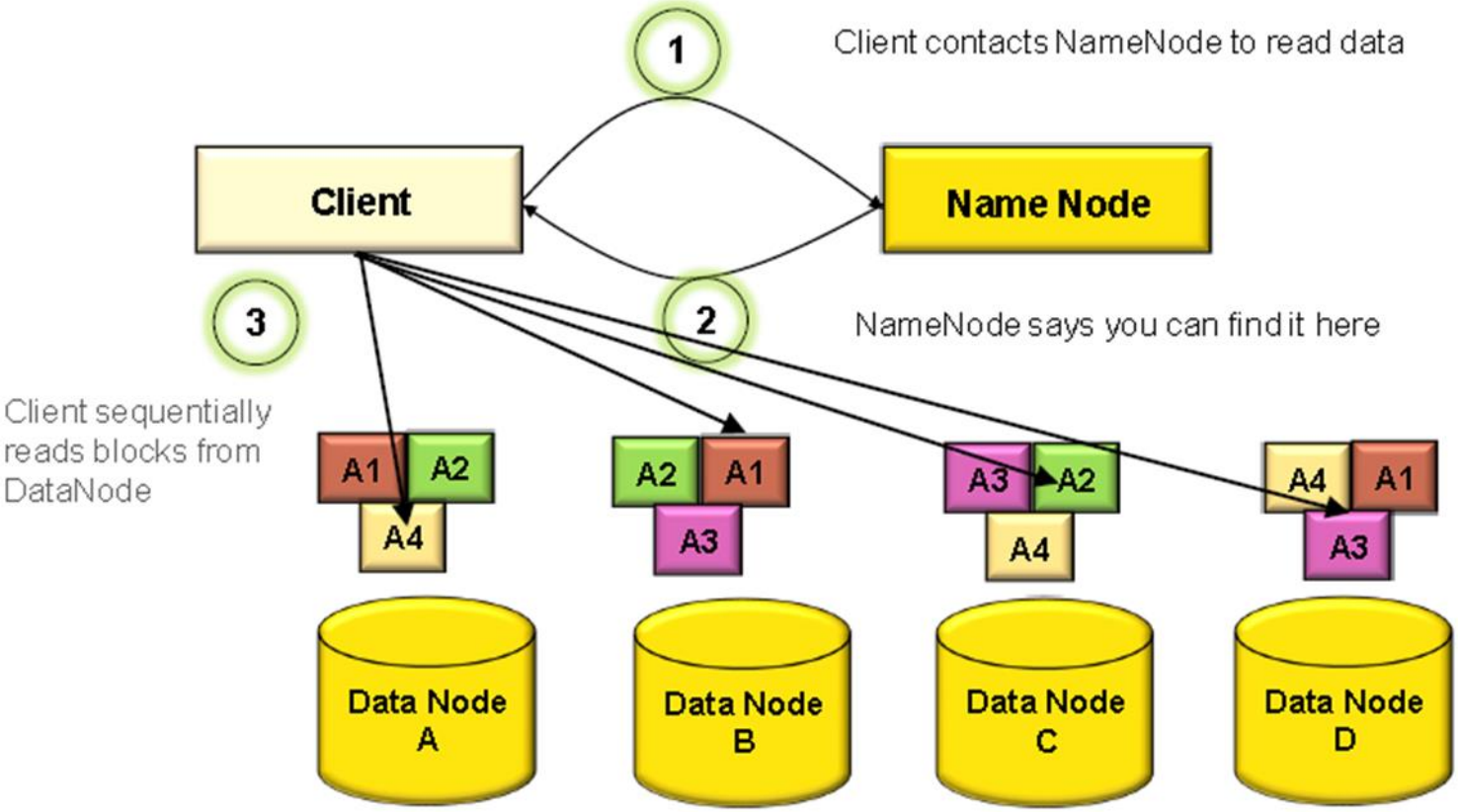
© Dr Zerabi S.

Haute disponibilité (High Availability)



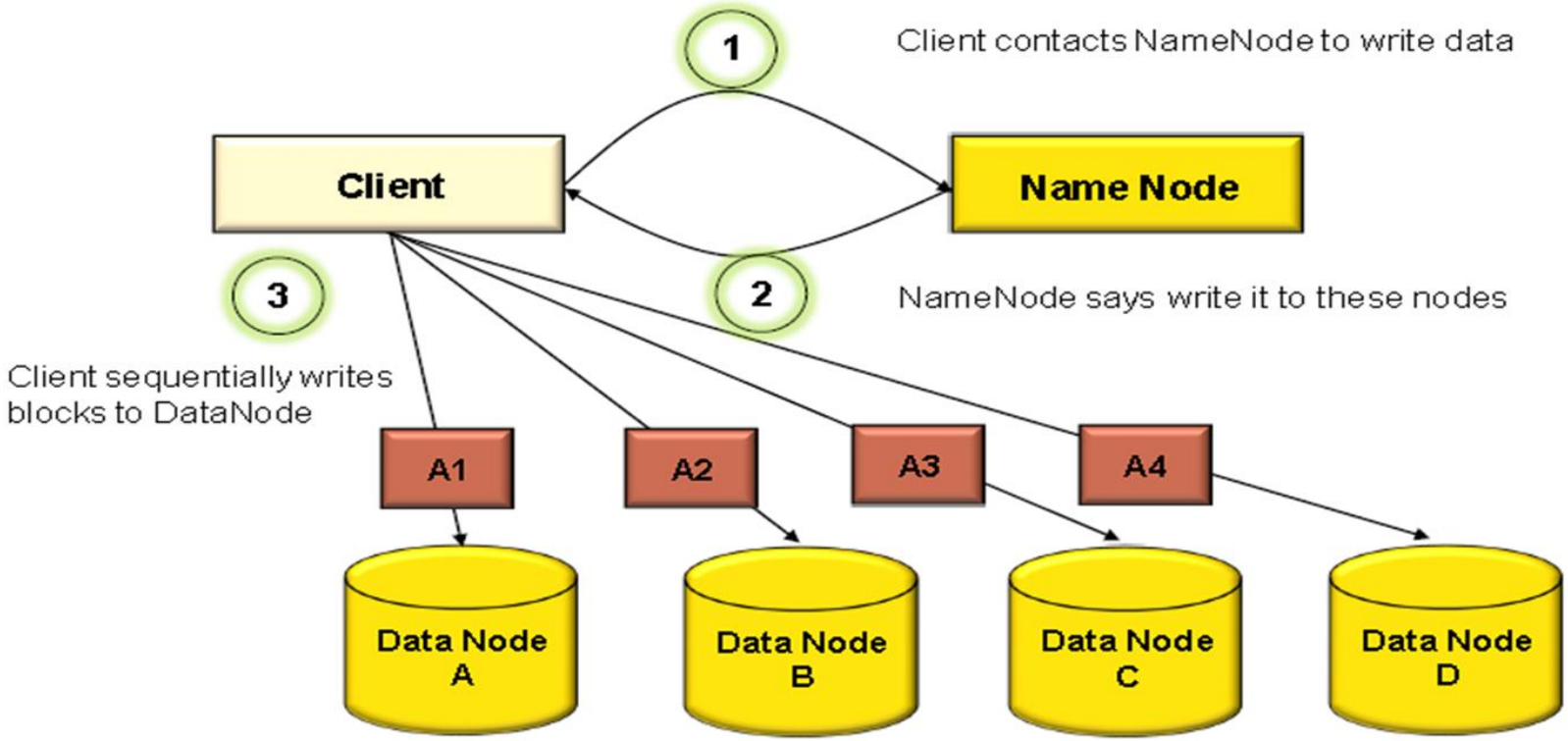
© Dr Zerabi S.

Opération de lecture



© Dr Zerabi S.

Opération d'écriture



© Dr Zerabi S.

MapReduce

- Un modèle de programmation qui fournit un cadre pour automatiser le calcul parallèle sur des données massives.
- Son principe consiste à découper les données en sous-ensembles de plus petite taille (lots ou fragments)
- Affecter chaque lot à une machine du cluster permettant ainsi leur traitement en parallèle.
- Agréger l'ensemble des résultats intermédiaires obtenus pour chaque lot pour construire le résultat final.
- l'ensemble des données est représentée sous la forme de paires (clé, valeur).
- Il utilise deux fonctions: **map()** et **reduce()**.

MapReduce

- **map** consiste à appliquer une même fonction à tous les éléments de la liste;

  map(f)[x0, ..., xn] = [f(x0, ...,f(xn)]

  map(*3)[2, 3, 6] = [6, 9, 18]

- **reduce** applique une fonction récursivement à une liste et retourne un seul résultat;

  reduce(f)[x0, ..., xn] = f(x0, f(x1, f(x2, ...)))

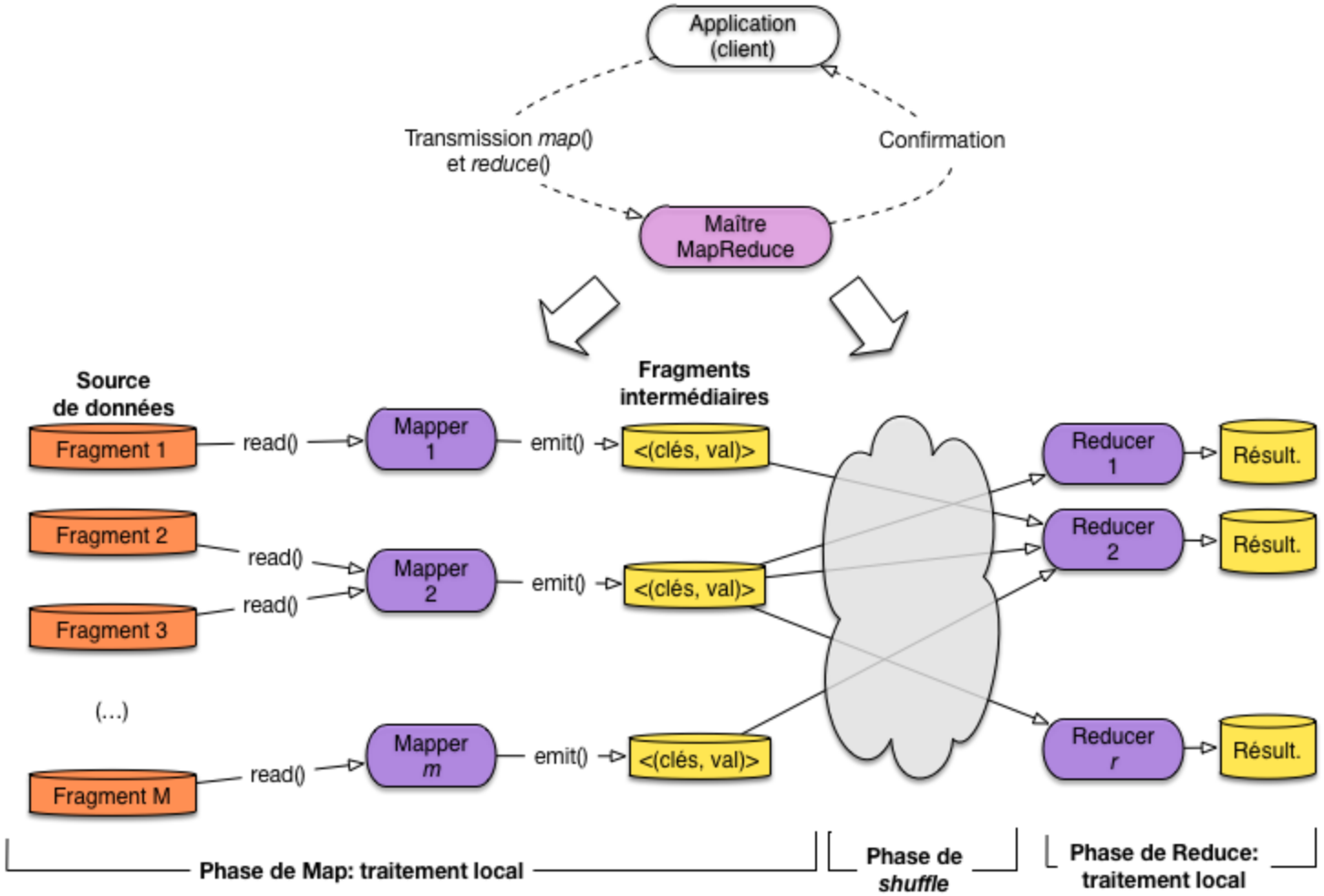
  reduce(+)[2, 3, 6] = (2 + (3 + 6)) = 11

© Dr Zerabi S.

Le fonctionnement de MapReduce

- L'ensemble des données à traiter est découpé en plusieurs lots ou sous-ensembles.
- Dans une première étape, la fonction map() est appliquée à chaque lot. Cette opération transforme la paire (clé, valeur) représentant le lot en une liste de nouvelles paires(clé, valeur)constituant ainsi des résultats intermédiaires du traitement à effectuer sur les données complètes.
- Avant d'être envoyés à l'étape REDUCE, les résultats intermédiaires sont regroupés et triés par clé. C'est l'étape de SHUFFLE (mélanger) and SORT (trier).
- Enfin, l'étape REDUCE consiste à appliquer la fonction reduce() à chaque clé. Elle agrège tous les résultats intermédiaires associés à une même clé et renvoie donc pour chaque clé une valeur unique.

MapReduce

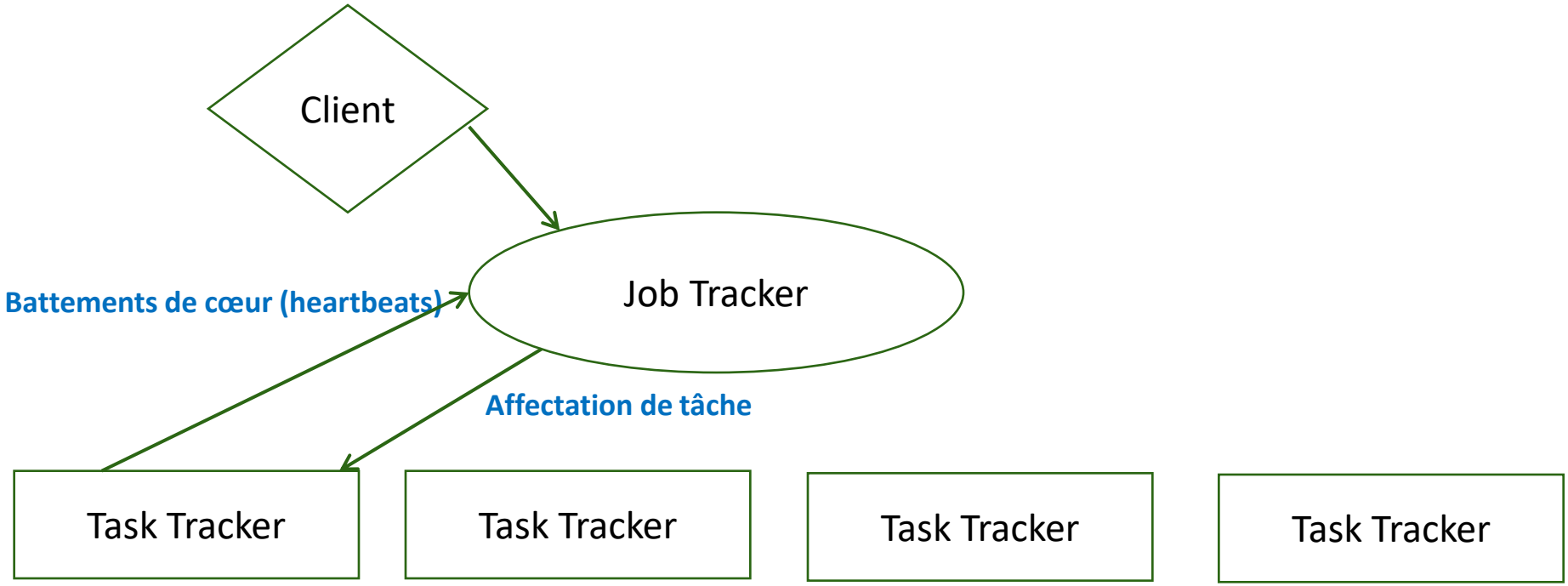


Fonctionnement de MapReduce

© Dr Zerabi S.

MapReduce (Hadoop 1.x)

- Dans Hadoop 1.x les composants de MapReduce suivent une architecture maître/esclave.
- **2 Daemons:** le Job Tracker (JT) et le TaskTracker (TT).

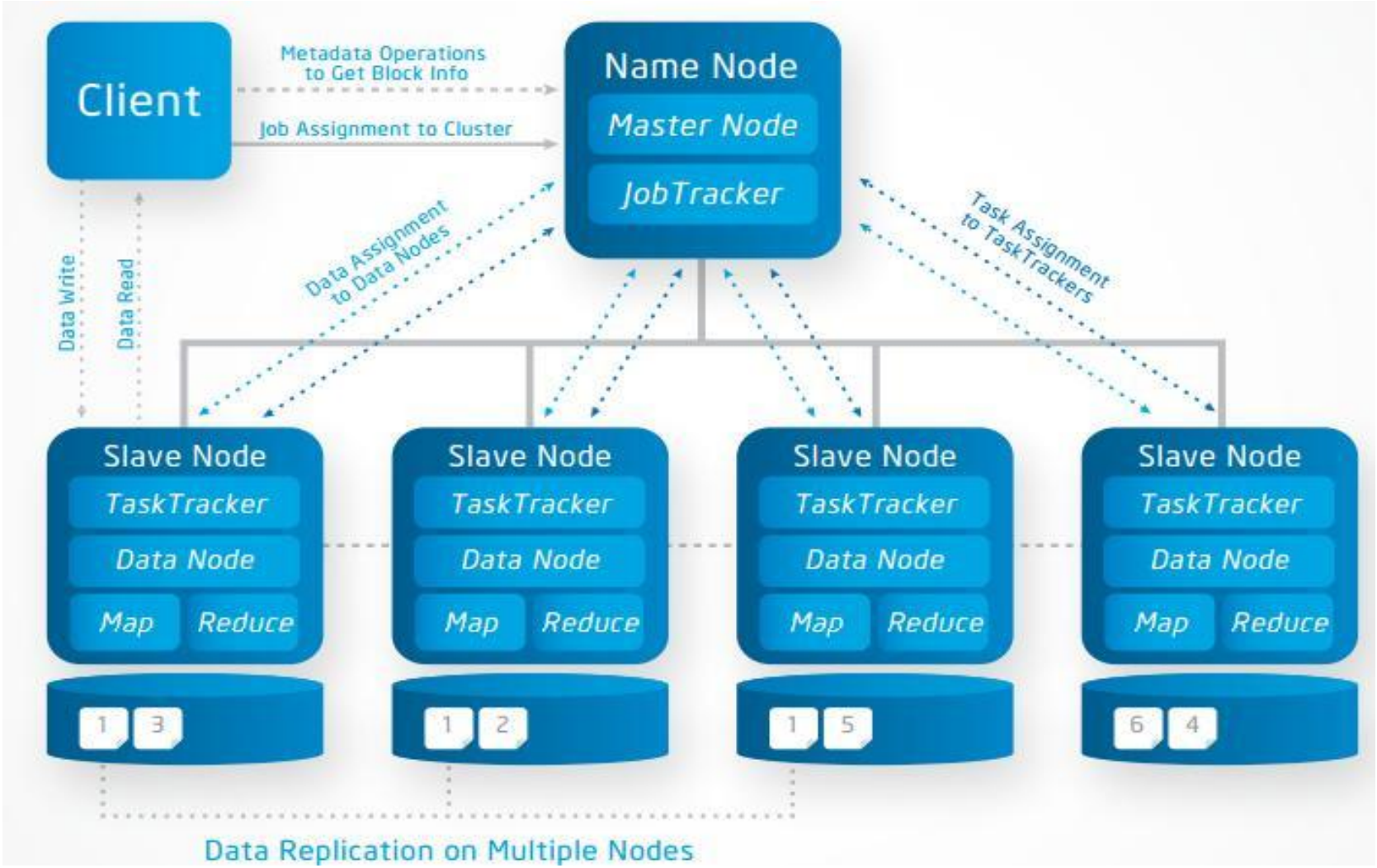


© Dr Zerabi S.

MapReduce (Hadoop1.x)

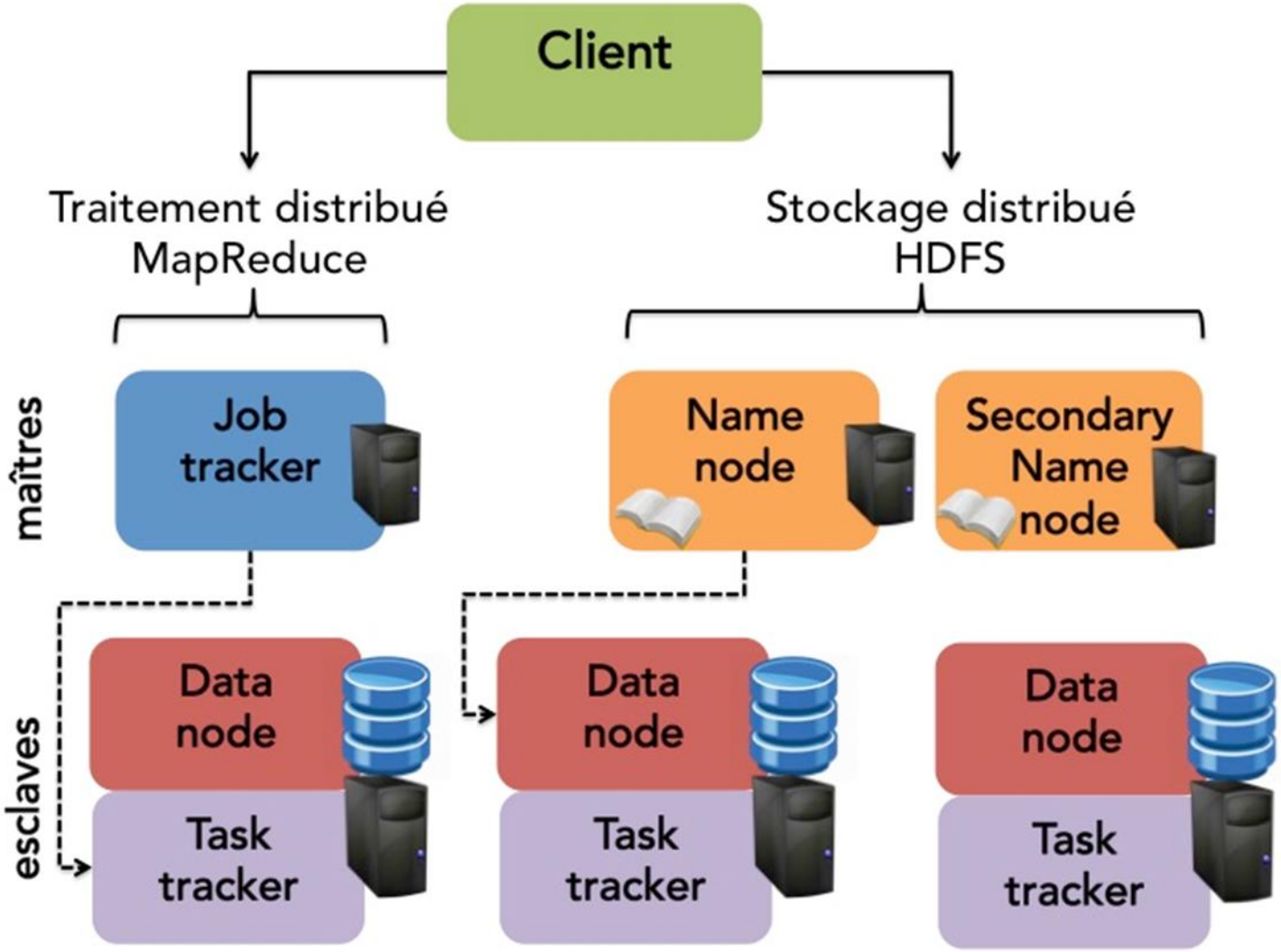
- Le JT permet de:
  - Savoir l'état du cluster et les nœuds actifs,
  - Reçoit les soumissions de jobs de Hadoop par les utilisateurs,
  - Assigne un identifiant à chaque job,
  - Distribue les tâches map et reduces aux différents nœuds.
- Le TT permet de:
  - Reçoit du JT des tâches à exécuter,
  - Lance l'exécution de chaque tâche,
  - Interroge periodiquement le JT par l'envoi à intervalle régulier les informations de service (heartbeats) au JT

MapReduce (Hadoop 1.x)

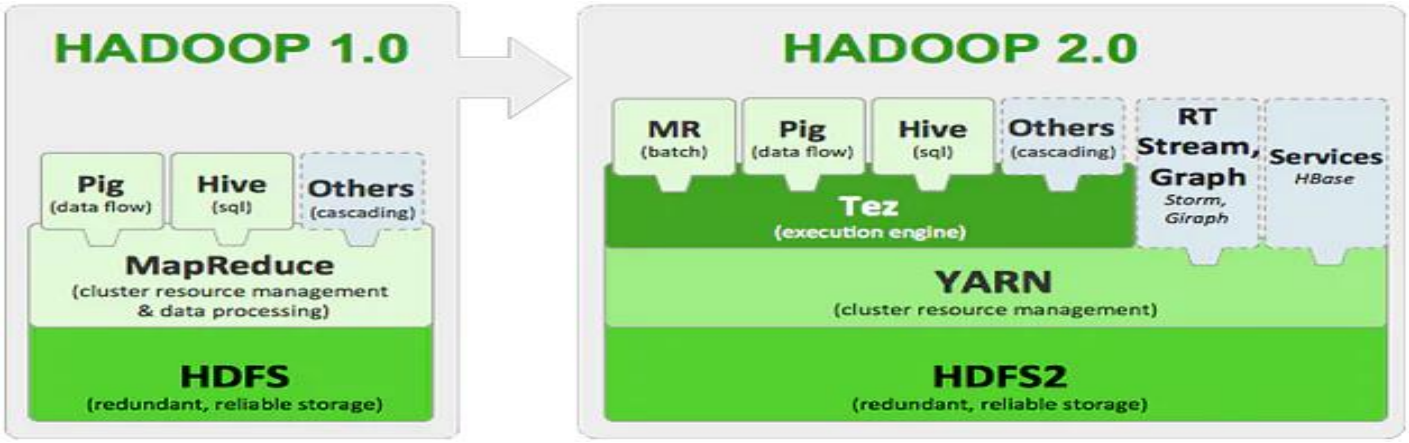


© Dr Zerabi S.

MapReduce (Hadoop 1.x)



© Dr Zerabi S.



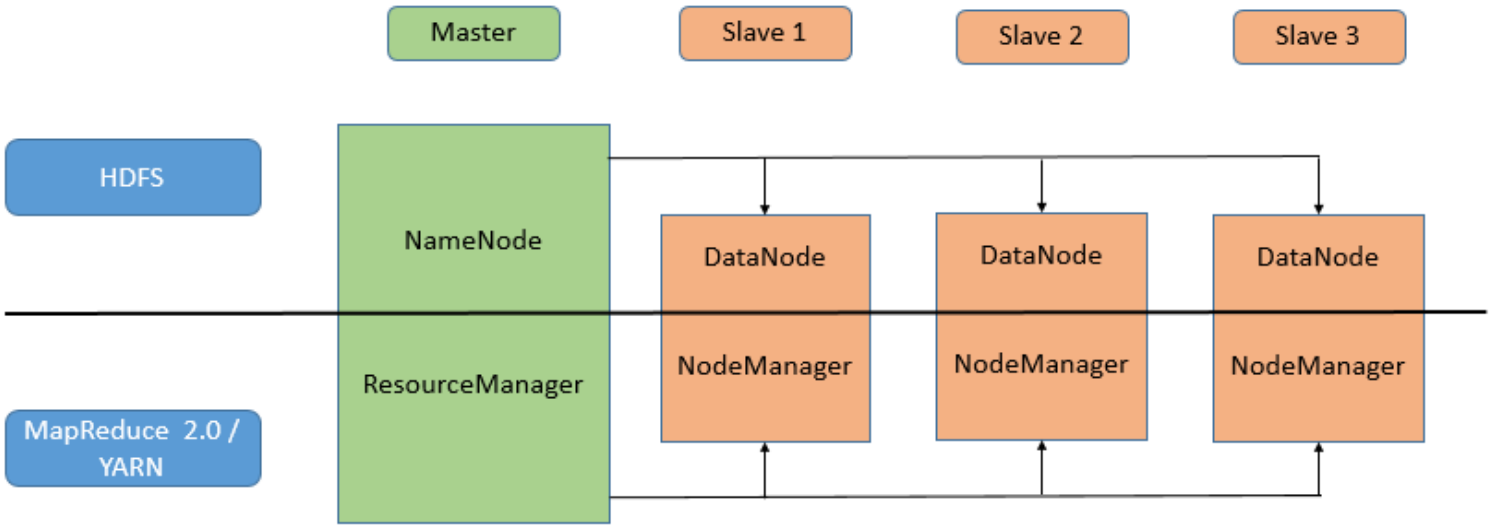
- La principale différence entre Hadoop **1.x** et Hadoop **2.x** est la **séparation entre la gestion des ressources du cluster** et **le modèle de traitement de données** dans la version 2 (assurée par **YARN)**.
- **YARN** (Yet Another Resource Negociator) permet de gérer les ressources du cluster.
- YARN permet par exemple de faire cohabiter dans un même cluster des jobs MapReduce et des jobs de traitement de graphes.

© Dr Zerabi S.

- YARN permet d'assurer une meilleure utilisation des ressources du cluster **en optimisant** l'allocation des nœuds map et des nœuds reduce.
- Dans Hadoop 1.x, la répartition des nœuds entre map et reduce est **figée** c.à.d. si un job cherche un nœud reduce et qu'il y a des nœuds map inutilisés, il est impossible de leur soumettre une tâche reduce.
- YARN permet aux utilisateurs de lancer des jobs MapReduce sur des données présentes dans HDFS et de suivre (monitor) leur avancement, récupérer les messages (logs) affichés par les programmes.
- YARN peut déplacer un processus d'une machine à l'autre en cas de défaillance ou d'avancement jugé trop lent.

© Dr Zerabi S.

YARN



© Dr Zerabi S.

YARN dispose des daemons suivants:

- **Le RessourceManager (RM):** (un par cluster)
  - c'est un daemon maître chargé de:
  - Allouer les ressources entre les différents jobs.
  - Optimiser l'utilisation du cluster en terme de disponibilité de ressources et les niveaux de priorité.
- **Le NodeManager (NM):** (un seul par nœud esclave)
  - c'est un daemon esclave
  - Il reçoit ses informations du RM puis gère l'affectation des ressources (cpu, RAM, ressources réseau, etc.) du nœud concerné entre les différents jobs qui s'exécutent sur le nœud.
  - Il informe le RM de l'utilisation des ressources du nœud.

© Dr Zerabi S.

Hadoop 1.x vs Hadoop 2.x vs Hadoop 3.x

	Hadoop1.x	Hadoop2.x	Hadoop3.x
Composants	-HDFS -MapReduce	-HDFS -Yarn -MapReduce	-HDFS -Yarn -MapReduce
Démons	-NameNode -DataNode -JobTracker -TaskTracker	-NameNode -DataNode -SecondaryNameNode -RessourceManager -NodeManager	-+eurs NameNode -DataNode -SecondaryNameNode -RessourceManager -NodeManager
performance	Moins performant	Plus performant (Yarn)	Encore plus performant
windows	Pas possible	possible	possible
Java		Min Java 7	Min Java 8
limites	Point de défaillance unique	Eliminer le Point de défaillance unique (NN actif+standBy)	Pas de Point de défaillance unique (+eurs NN actifs)

© Dr Zerabi S.

Hadoop v1 vs Hadoop v2 vs Hadoop v3

	Hadoop1.x	Hadoop2.x	Hadoop3.x
Tolérance aux pannes	Réplication classique	Réplication classique	Erasure coding (codage d'effacement) fragments de parité
Haute disponibilité du système	minimale	moyenne	maximale
Maintenance NameNode	Maintenance manuelle	Maintenance manuelle	Maintenance automatique
Coût de stockage	HDFS consomme 300% d'espace de stockage	HDFS consomme 300% d'espace de stockage	HDFS consomme 150% d'espace de stockage
Scalabilité	<4.000 nœuds	<10.000 nœuds	>10.000 nœuds

© Dr Zerabi S.