

10

# Hadoop: Présentation et écosystème



# Présentation du Framework

11

- Hadoop est un framework **open source, écrit en Java** et géré par la fondation Apache.
- **Java est le langage de préférence** pour écrire des programmes Hadoop natifs. Néanmoins, il est possible d'utiliser Python, Bash, Ruby, Perl ...
- Hadoop offre un **rapport performance-prix très compétitif** (Amazon EMS, réutilisation de PC existants, aucun coûts de licences ni de matériel spécialisé HPC).
- Le nom “Hadoop” était initialement celui d'un éléphant en peluche, jouet préféré du fils de **Doug Cutting**.

Doug Cutting créateur de Hadoop.

BigData



# Présentation du Framework

12

□ Le projet Hadoop consiste en deux grandes parties:

□ Stockage des données : **HDFS (Hadoop Distributed File System)** 

□ Traitement des données : **MapReduce / Yarn**



□ **Principe:**

□ Diviser les données

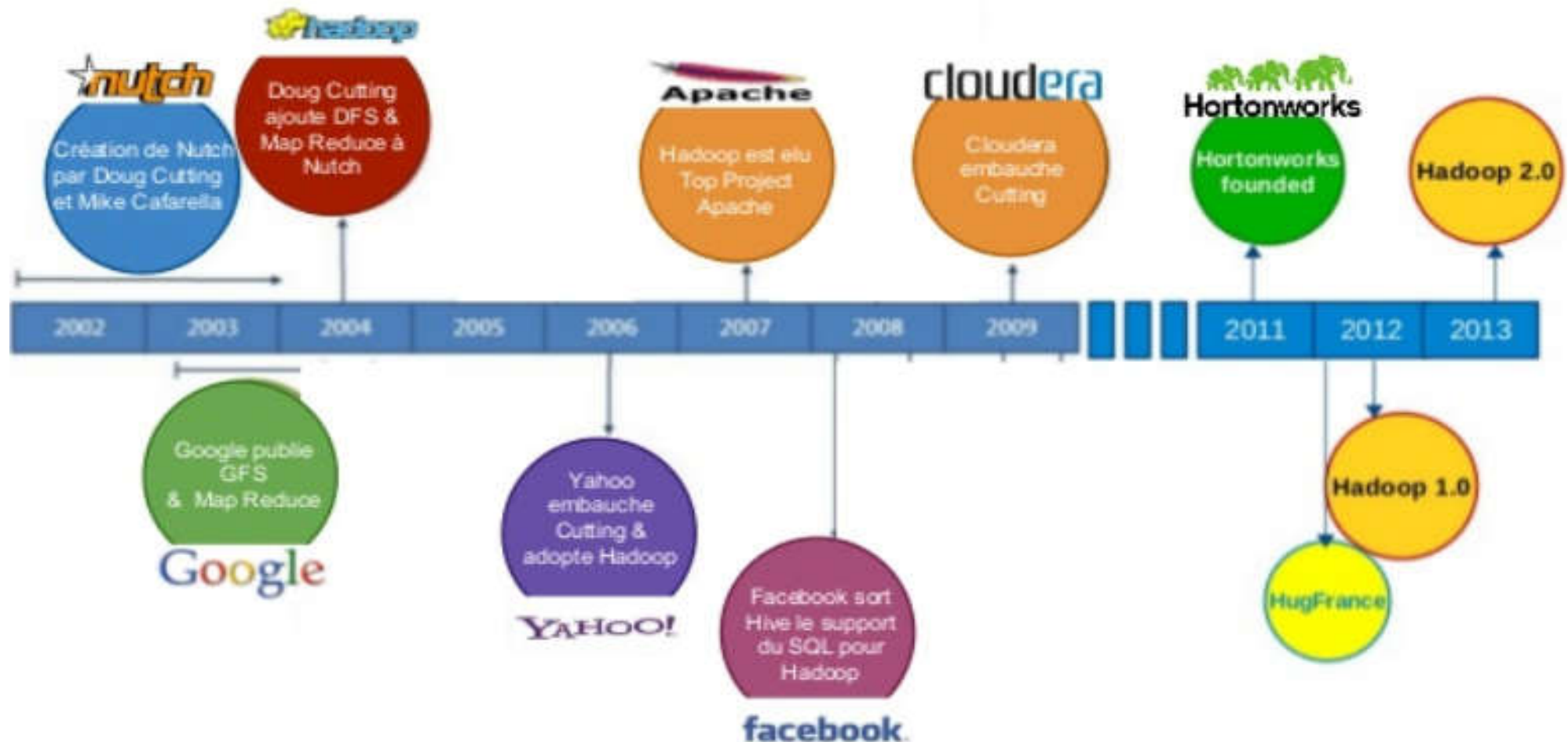
□ Les sauvegarder sur une collection de machines, appelée cluster

□ Traiter les données directement là où elles sont stockées, plutôt que de les copier à partir d'un serveur distribué

□ Il est possible d'ajouter des machines à votre cluster, au fur et à mesure que les données augmentent. Les machines peuvent être hétérogènes.

# Historique de Hadoop

13



# Qui utilise Hadoop ?

14



facebook

IBM

The New York Times

JPMorganChase

eHarmony

twitter



NETFLIX

rackspace  
HOSTING

amazon.com



NING

SAMSUNG

YAHOO!

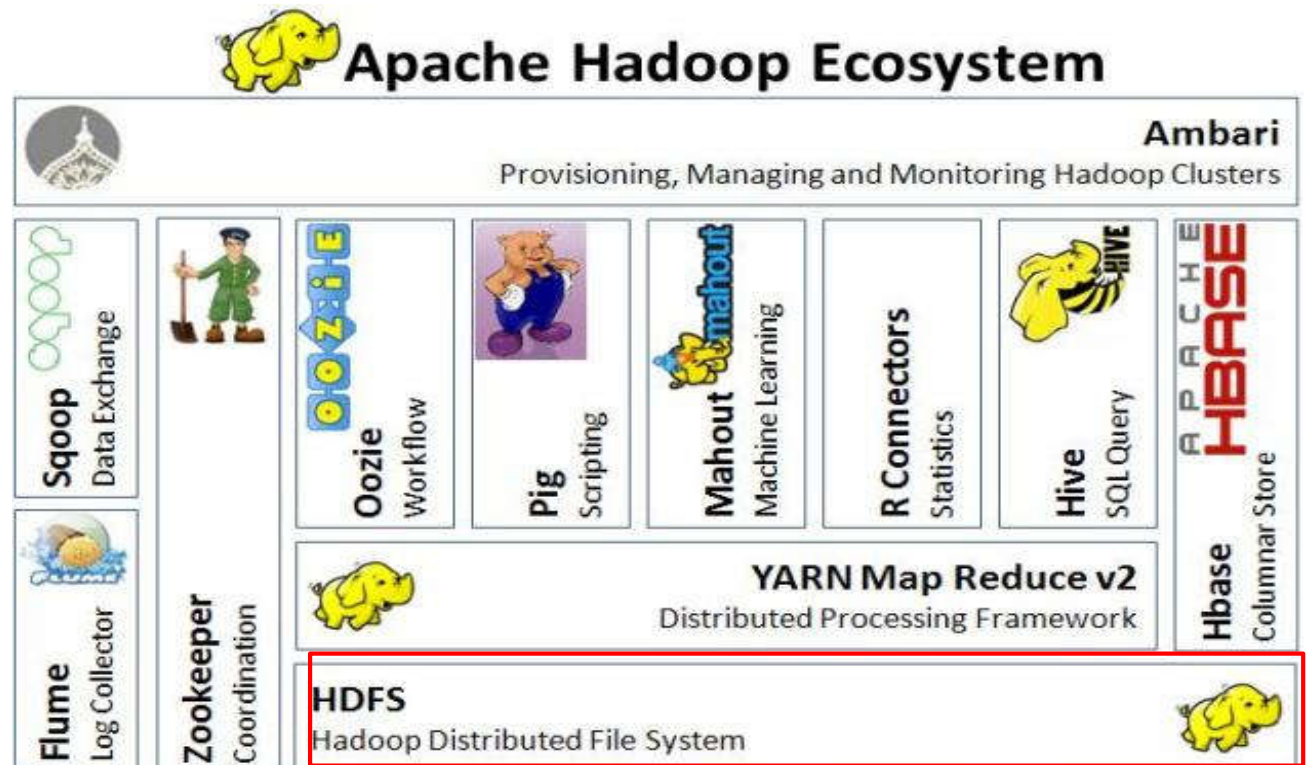
BigData

# Ecosystème de Hadoop

15

- Le système de fichiers distribués Hadoop (HDFS) est basé sur le système de fichiers Google (GFS).
- Il fournit un système de fichiers distribué conçu pour fonctionner sur du matériel standard et présente de nombreuses similitudes avec les systèmes de fichiers distribués existants avec certaines différences.
- Il est hautement tolérant aux pannes et est conçu pour être déployé sur du matériel à faible coût.

NB: La liste présentée ci-dessus n'est pas exhaustive!

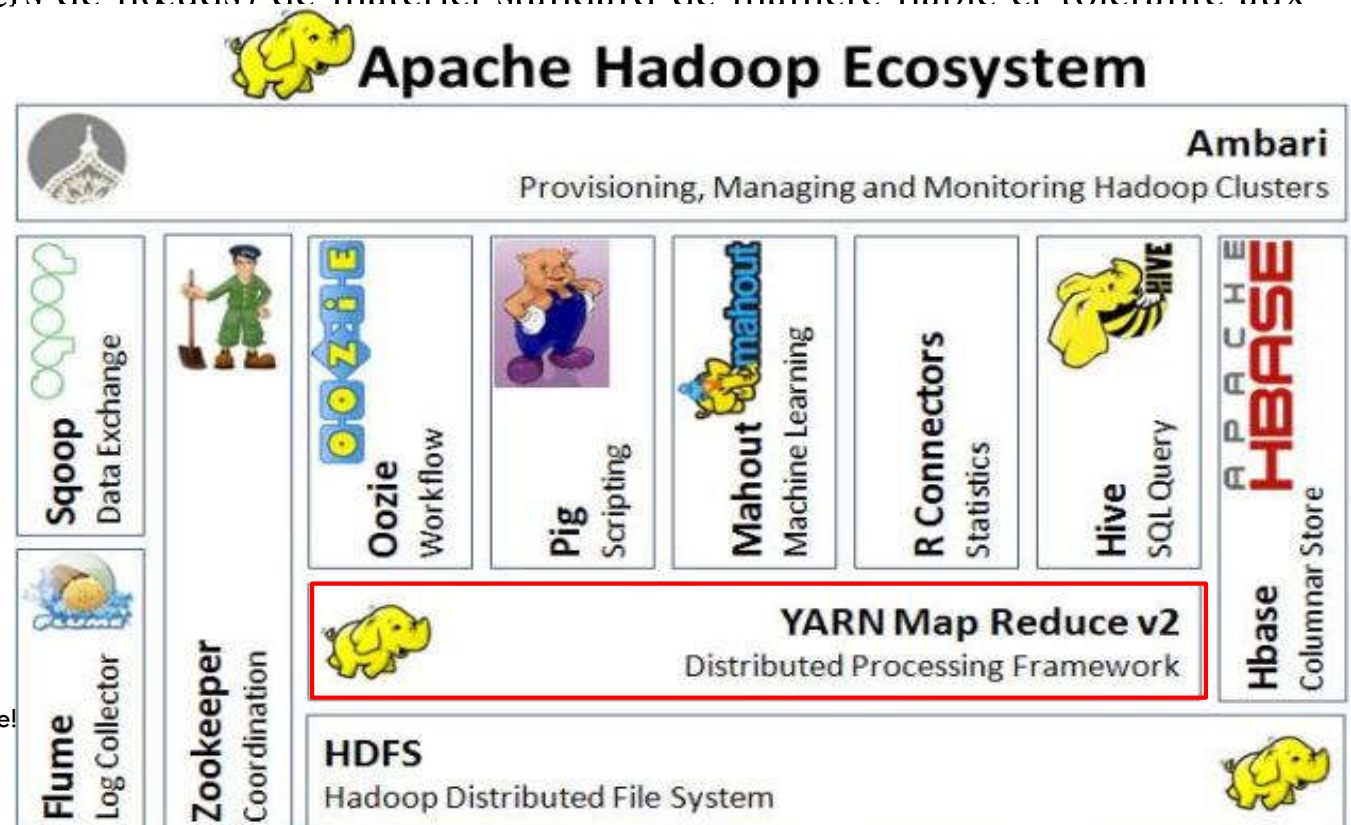




# Ecosystème de Hadoop

16

- MapReduce est un modèle de programmation parallèle pour l'écriture d'applications distribuées conçu par Google.
- Permet un traitement efficace de grandes quantités de données (plusieurs téraoctets) sur des grappes étendues (des milliers de nœuds) de matériel standard de manière fiable et tolérante aux pannes.



NB: La liste présentée ci-dessus n'est pas exhaustive!

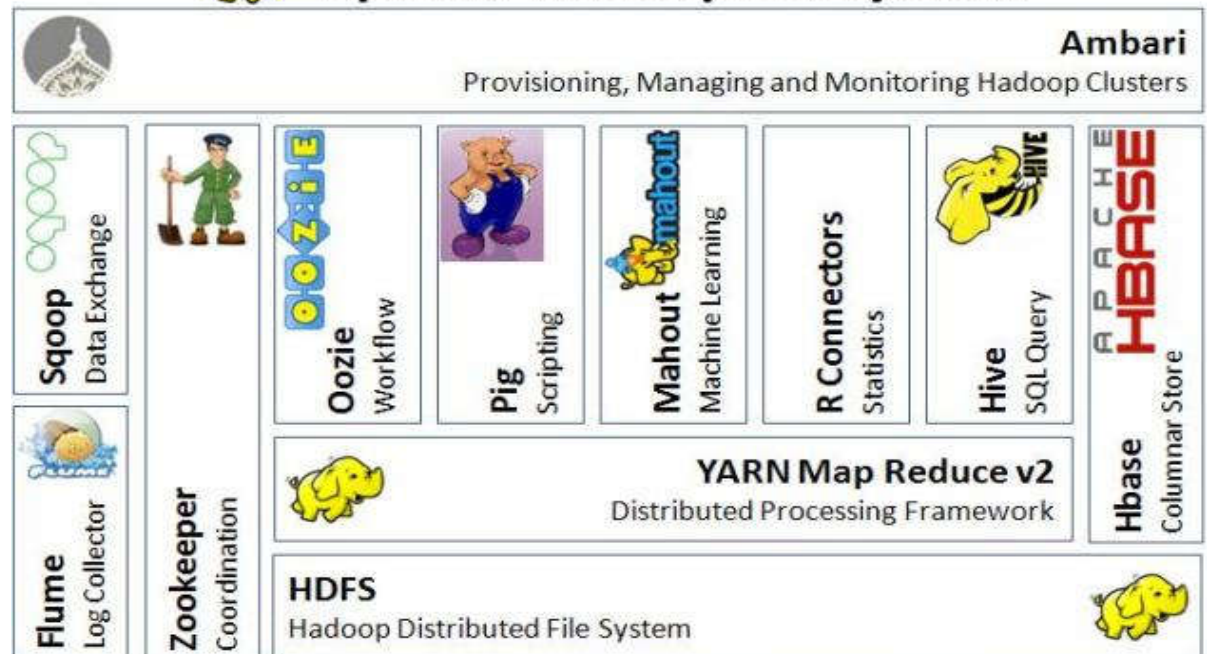
# Ecosystème de Hadoop

17

- Plusieurs outils existent pour permettre:
  - L'extraction et le stockage des données de/sur HDFS
  - La simplification des opérations de traitement sur ces données
  - La gestion et coordination de la plateforme
  - Le monitoring du cluster



## Apache Hadoop Ecosystem



NB: La liste présentée ci-dessus n'est pas exhaustive!





# HDFS : Hadoop Distributed File System

## - Définition-

19

- ❑ HDFS est **un système de fichiers distribué** et la couche native **de stockage et d'accès** à des données d'Hadoop.
- ❑ Conçu pour stocker et gérer des fichiers de **très grande taille** dans un cadre distribué de manière **transparente** comme s'ils étaient sur le disque dur local.
- ❑ HDFS est **hautement tolérant aux pannes** et conçu avec un **matériel à faible coût**.
- ❑ Pour garantir une tolérance aux pannes, les blocs de chaque fichier sont **répliqués**, de manière intelligente, sur plusieurs machines.

# HDFS : Hadoop Distributed File System

## - Types des noeuds-

20

- 2 types de noeuds dans HDFS:
  - ▣ 1 seul noeud pour les **méta-données** : les **noms** et **blocs des fichiers** ainsi que **leur localisation** dans le cluster (**un gros annuaire**).
  - ▣ plusieurs noeuds pour sauvegarder les données réelles

### Data Node:

Exemple de configuration du Name Node

Processors: 2 Quad Core CPUs running @ 2 GHZ

RAM: 128 GB

Disk: 6 x 1TB SATA

Network: 10 Gigabit Ethernet

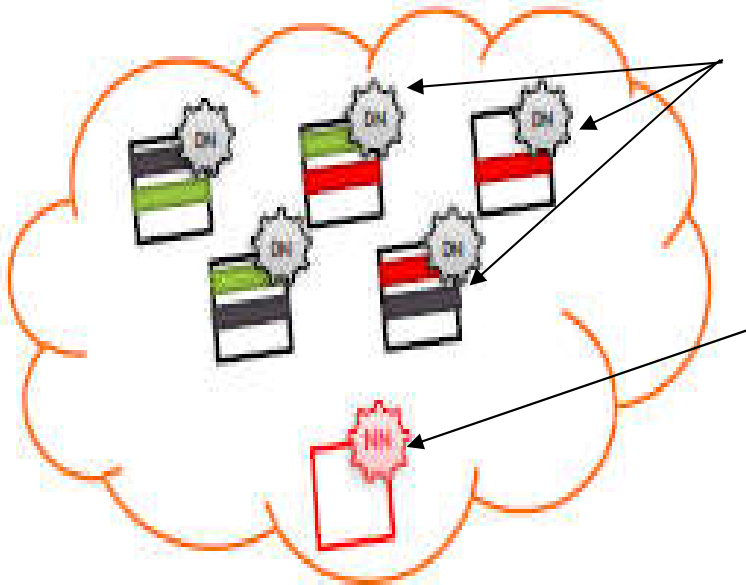
### Name Node

Exemple de Configuration du Data Node

Processors: 2 Quad Core CPUs running 4 2 GHz

RAM: 64 GB

Disk: 12-24 x 1TB SATA Network: 10 Gigabit Ethernet

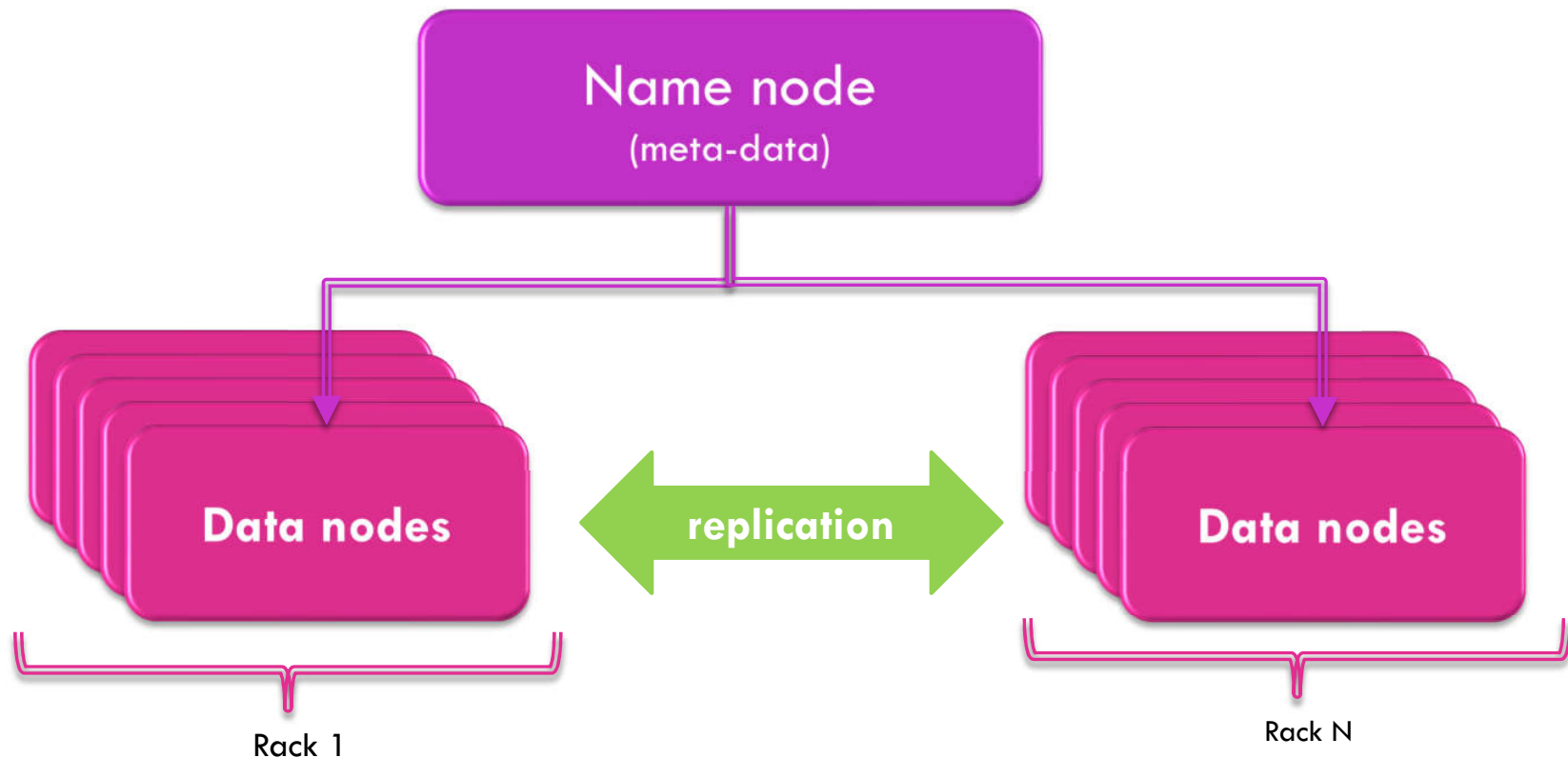


# HDFS: Hadoop Distributed File System

## - Architecture Hadoop 1.0 -

21

### □ Topologie Maître-esclave



BigData

# HDFS : Hadoop Distributed File System

## - Types des noeuds-

22

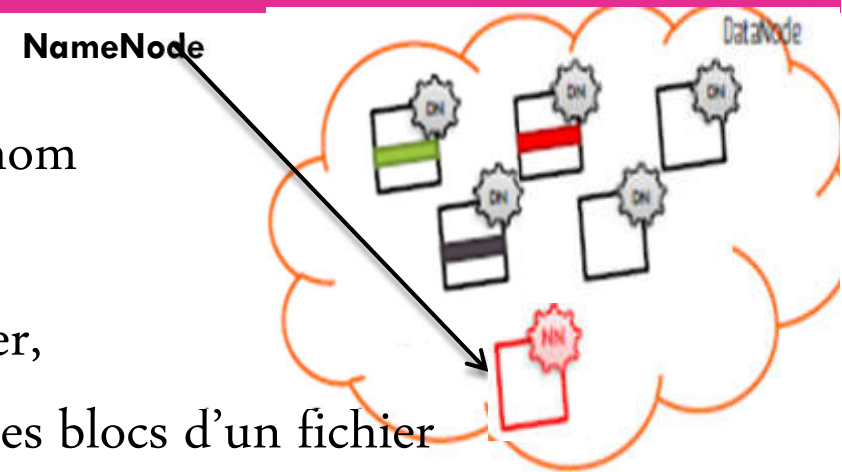
### ☐ NameNode

- ☐ Une machine séparée sur lequel tourne un dénom appelé namenode
- ☐ Contient des métadonnées, pas de données user,
- ☐ Permet de retrouver les nœuds qui hébergent les blocs d'un fichier
- ☐ Cordonne l'accès aux données dans le système de fichiers.
- ☐ La responsabilité principale d'un Namenode est de stocker le namespace de HDFS :

- L' arborescence des répertoires
- Les permissions des fichiers
- Le mapping des fichiers aux blocs

- ☐ Ces données sont stockées dans deux structures : **FsImage** et **EditLog**

BigData



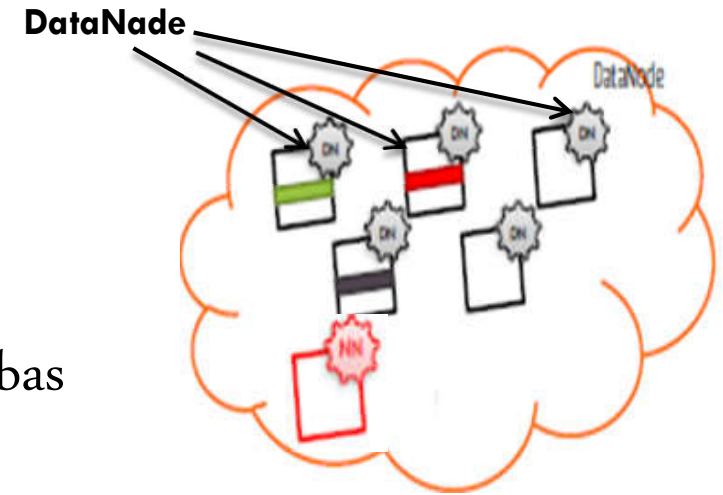
# HDFS : Hadoop Distributed File System

## - Types des noeuds-

24

### ☐ DataNode (nœud de données)

- ☐ Une machine sur laquelle tourne un démon esclave
- ☐ Il stocke les données réelles.
- ☐ Exécute les demandes de lecture et d'écriture de bas niveau des clients du système de fichiers.



- ☐ Il y a un DataNode **pour chaque machine** au sein du cluster.
- ☐ Les Datanodes sont sous les ordres du Namenode et sont surnommés les **Workers (ou esclaves)**.

➔ Ils sont donc sollicités par les Namenodes lors des opérations de lecture et d'écriture!

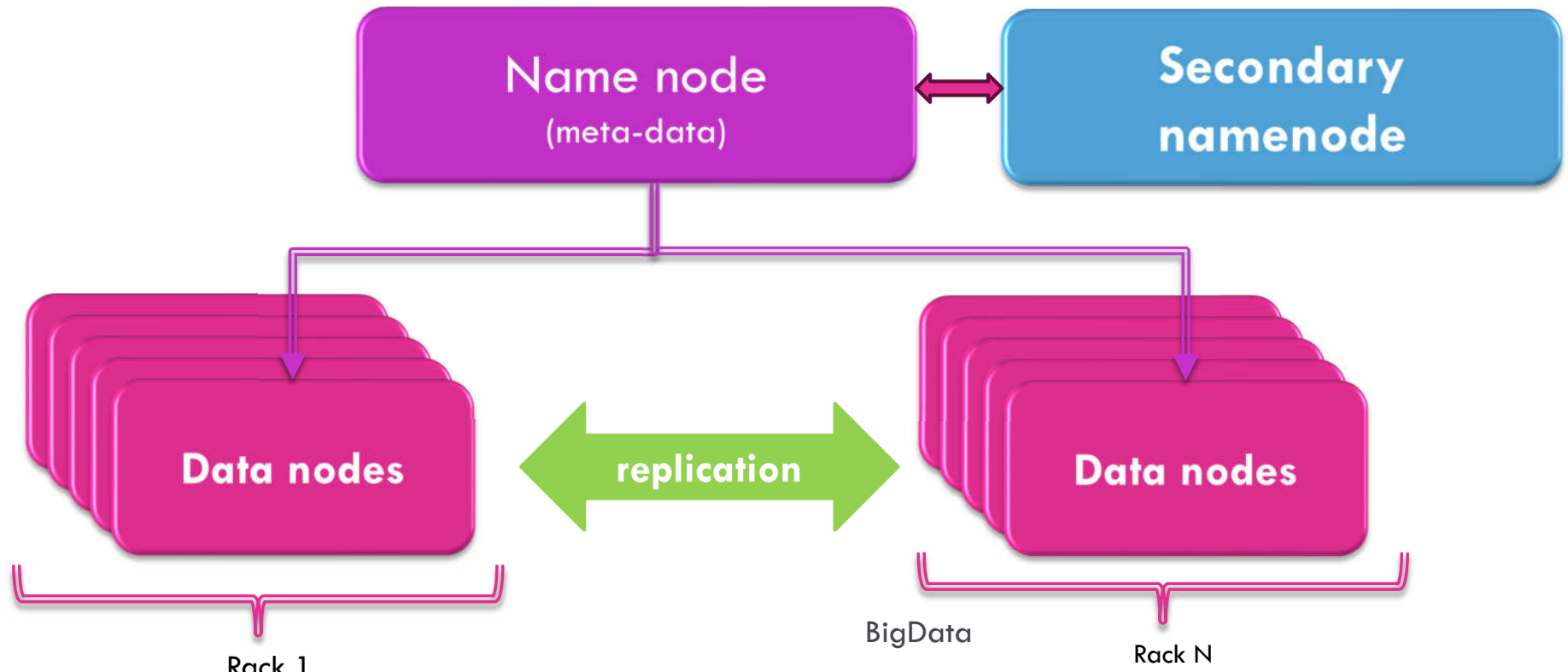


# HDFS: Hadoop Distributed File System

## - Architecture Hadoop 2.0 -

25

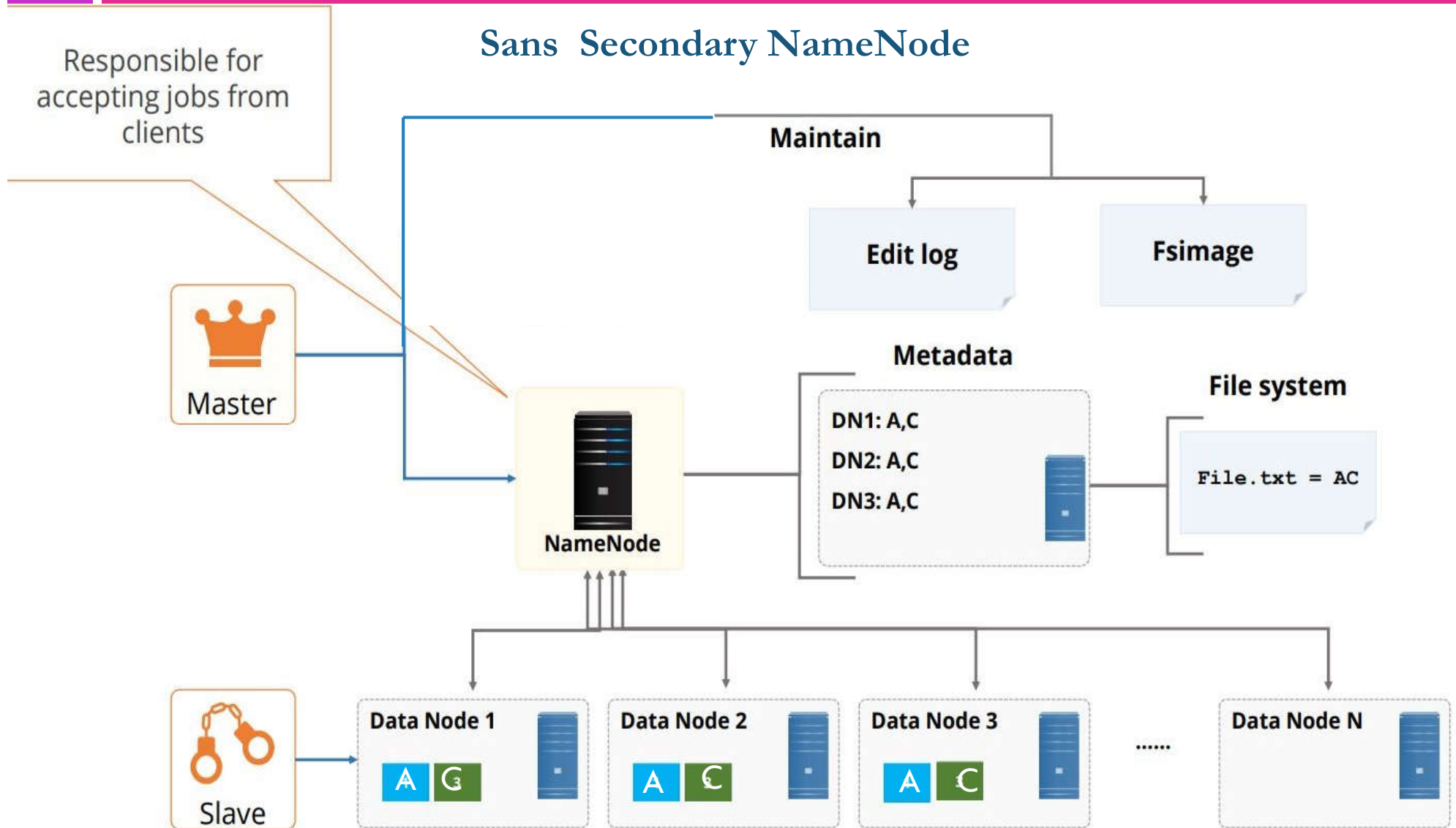
- **Problème** : Trop de charge sur le NameNode
- **Solution**: Ajout du **Secondary NameNode**: un assistant du NameNode pour réduire sa charge ( Ne le remplace pas)



# HDFS: Hadoop Distributed File System

## - Rôle du Secondary Namenode-

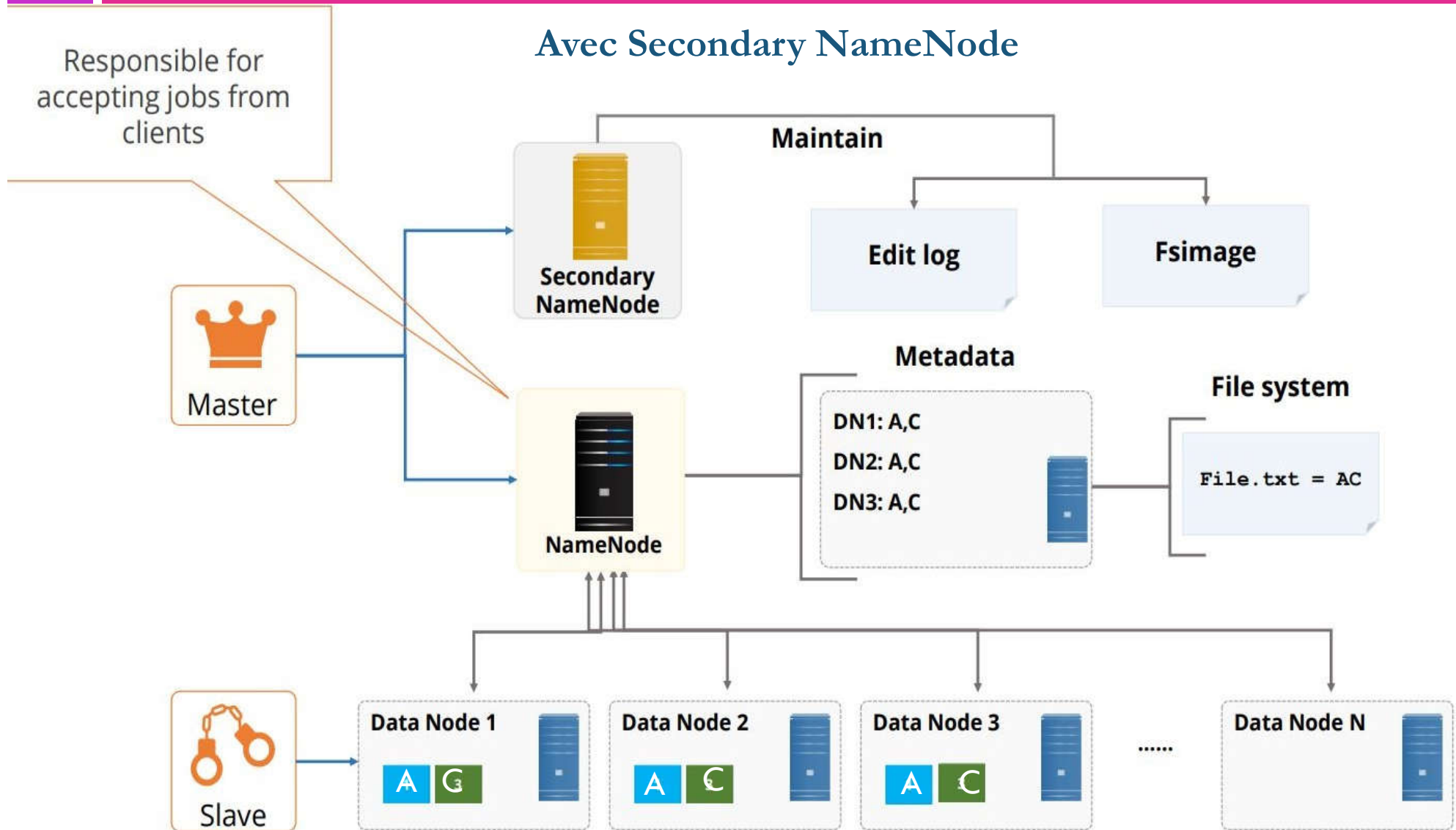
26



# HDFS: Hadoop Distributed File System

## - Rôle du Secondary Namenode-

27

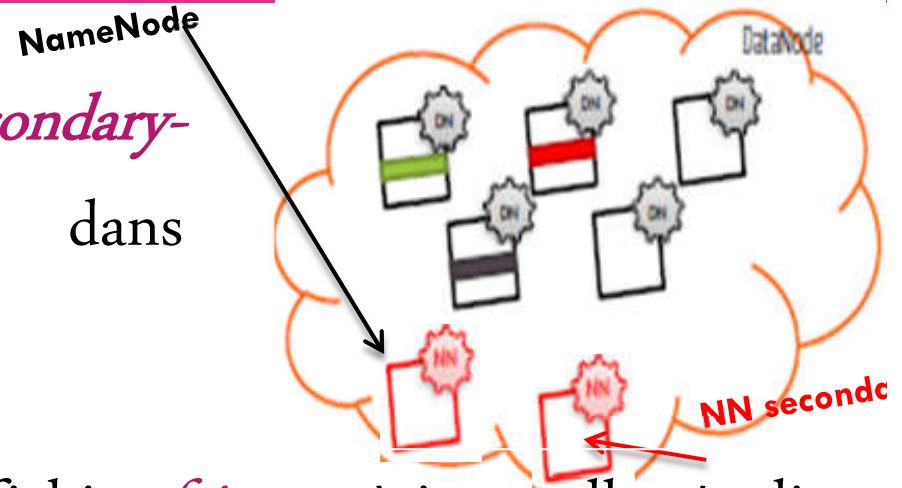


# HDFS: Hadoop Distributed File System

## - Rôle du Secondary Namenode-

28

- ❑ Ce Nœud supplémentaire appelé *Secondary-Namenode* a été mis en place dans l'architecture Hadoop version2.
- ❑ le *Secondary NameNode* met à jour le fichier *fsimage* à intervalle régulier en y intégrant le contenu des *editlogs*.
- ❑ Le *SecondaryNameNode* intervient soit :
  - lorsque le fichier *editlogs* atteint une taille prédéfinie.
  - à intervalle régulier (par exemple une fois par heure).

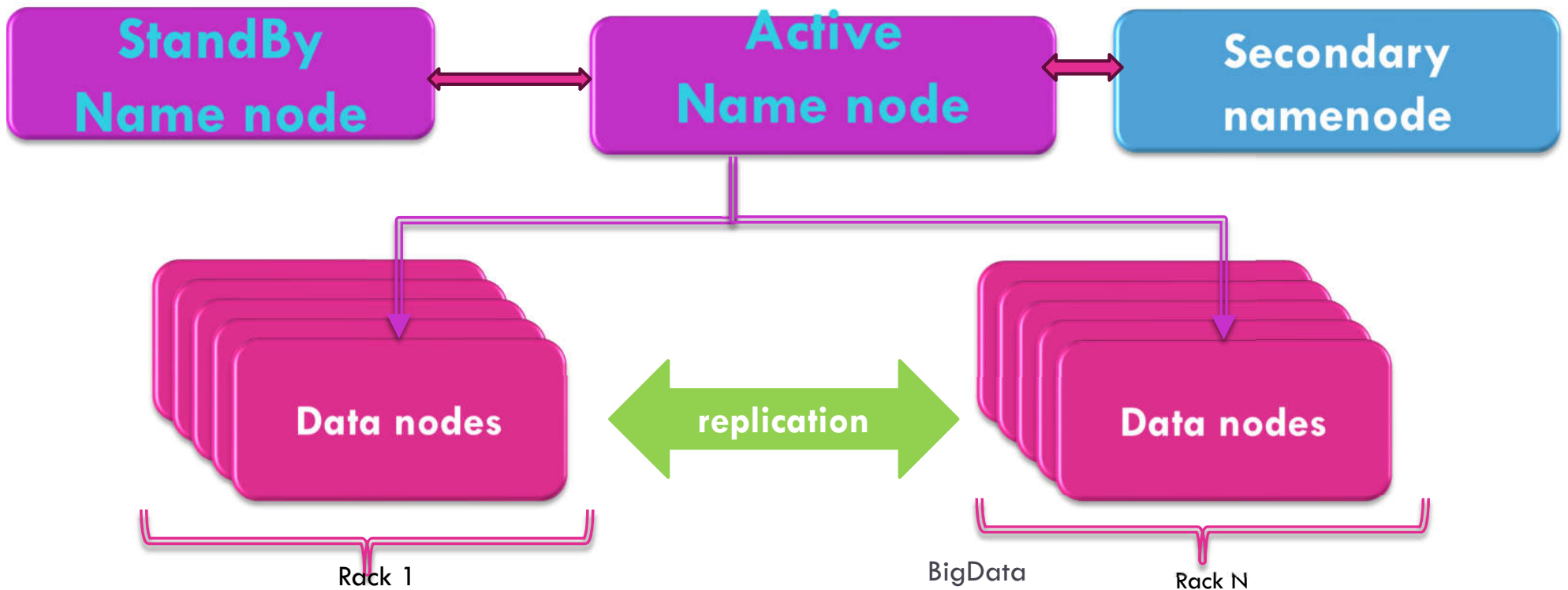


# HDFS: Hadoop Distributed File System

## - Architecture Hadoop 2.0 -

29

- ❑ **Problème** : Le Namenode est un **SPOF** (Single Point Of Failure): Si ce service est arrêté, le cluster sera non disponible!
- ❑ **Solution**: Possibilité de mettre plusieurs instances du NameNode (un **actif** et un ou plusieurs **StandBy** : solution **High Availability (HA)** mais plus coûteuse).



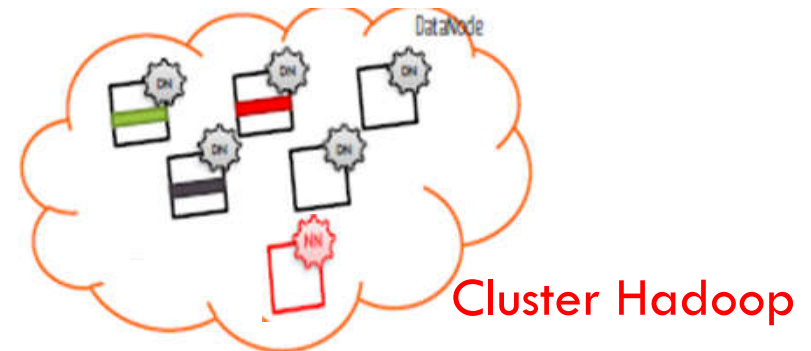
# HDFS: Hadoop Distributed File System

## -Décomposition en blocs-

30

- ❑ Les fichiers sont physiquement **découpés en blocs d'octets de grande taille** (64 Mo: Hadoop 1.x et 128 Mo (Hadoop 2.x) ) pour optimiser les temps de transfert et d'accès
- Quand un fichier mydata.txt est enregistré dans HDFS, il est décomposé en grands blocs chaque bloc ayant un nom unique: blk\_1, blk\_2...

mydata.txt (150 Mo)



- ❑ Ces blocs sont ensuite **répartis** sur plusieurs machines, permettant ainsi de traiter **un même fichier en parallèle**.
- ❑ Cela permet aussi de ne pas être limité par **la capacité de stockage** d'une seule machine pour au contraire tirer parti de tout l'espace disponible du cluster de machines ;



# HDFS: Hadoop Distributed File System

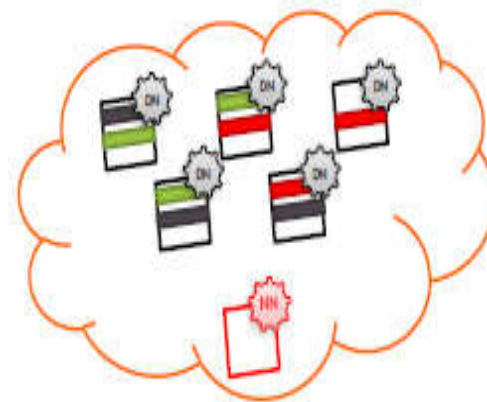
## -Réplication des données -

31

Si l'un des nœuds a un problème, les données seront perdues ?

- ❑ Hadoop réplique chaque bloc 3 fois (par défaut)
- ❑ Le facteur de réplication par défaut peut être ajusté en modifiant la propriété *dfs.replication* dans le fichier de configuration *hdfs-site.xml*.

mydata.txt (150 Mo)



Cluster Hadoop

- ❑ Si un nœud tombe en panne, les blocs qui y étaient hébergés seront répliqués pour avoir toujours 3 copies stockées.
- ❑ Application du concept de **Rack-Awareness** dans la réplication

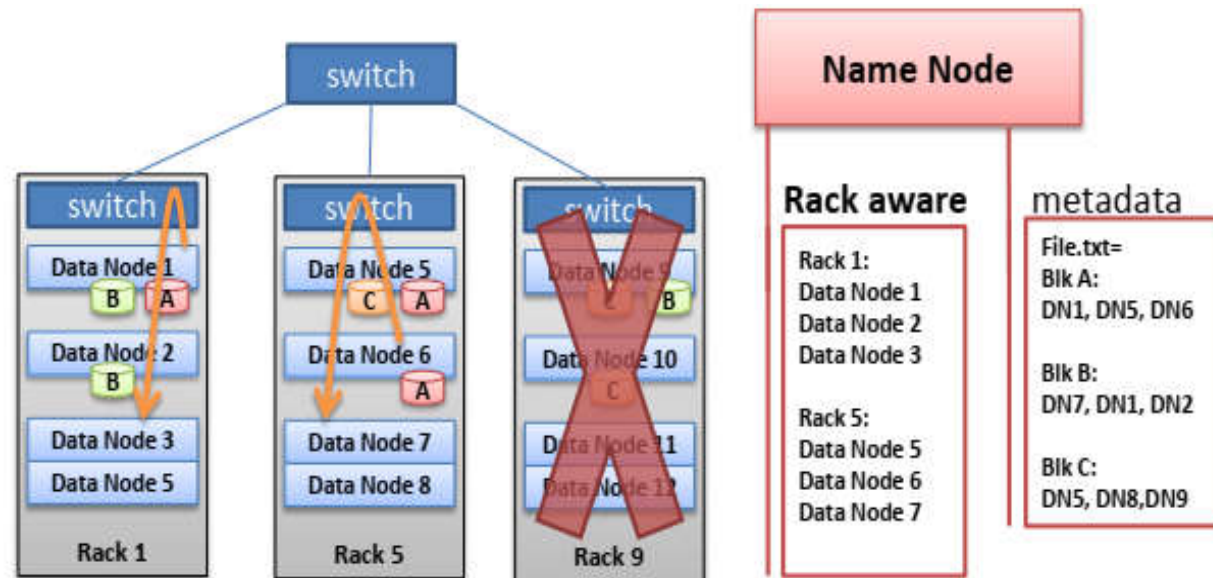
BigData

# HDFS: Hadoop Distributed File System

## - Principe du Rack Awareness -

32

- ❑ 2 copies de chaque bloc se trouvent dans deux nœuds de données distincts du même rack afin de réduire la latence,
- ❑ La troisième copie est placée sur un autre rack pour améliorer la redondance et la disponibilité.

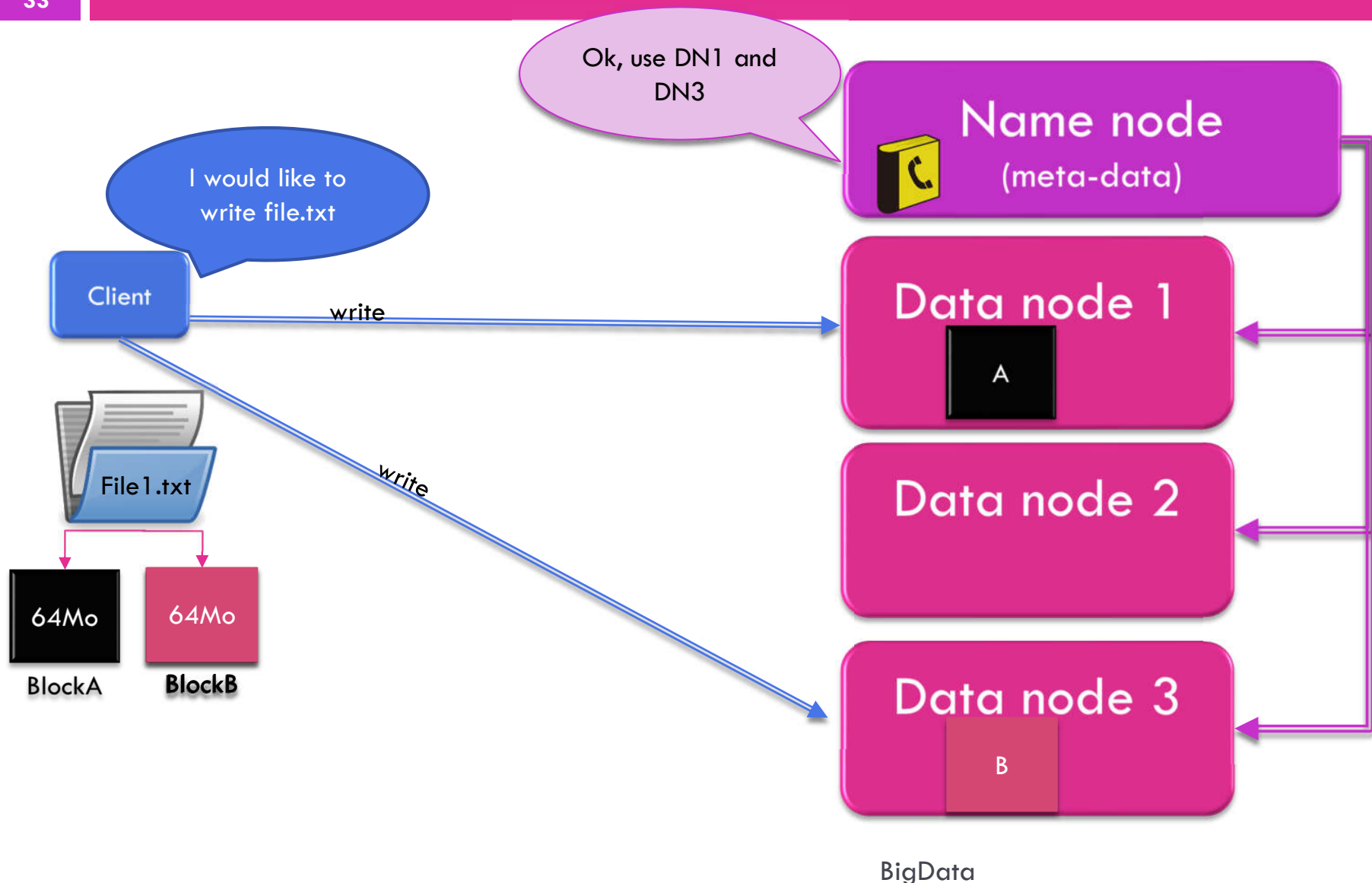


- Ne jamais perdre toutes les données si tout le rack tombe en panne.
- Gardez les flux volumineux dans le rack lorsque cela est possible.
- Répartition basée sur l'hypothèse que dans le rack la bande passante est meilleure et la latence est faible.

# HDFS: Hadoop Distributed File System

## Ecriture/Lecture d'un fichier

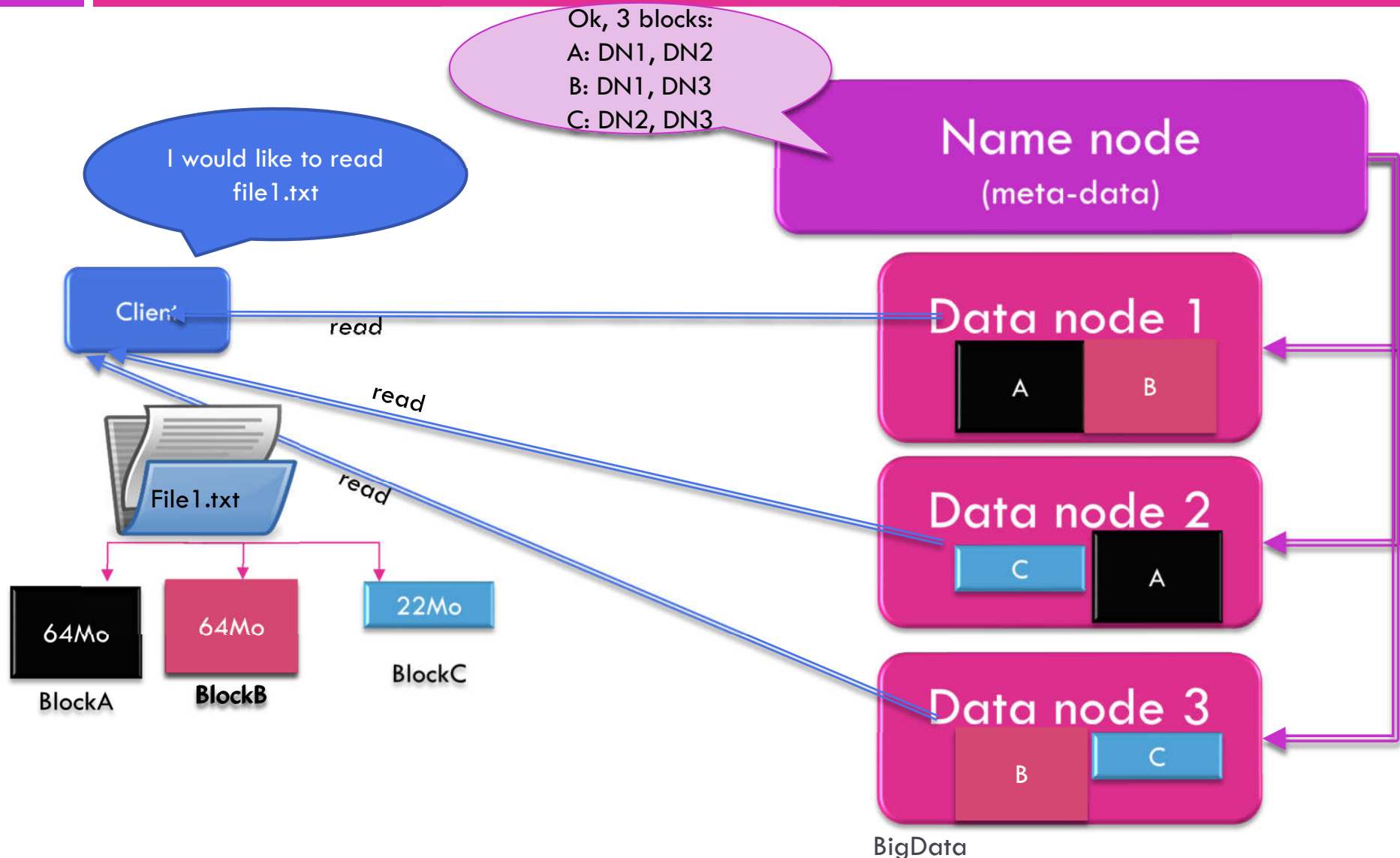
33



# HDFS: Hadoop Distributed File System

## Ecriture/Lecture d'un fichier

34



# HDFS: Hadoop Distributed File System

## Ecriture/Lecture d'un fichier

35

- HDFS suit la philosophie **Write Once – Read Many**.
- Il n'est pas possible d'éditer un fichier existant déjà dans HDFS.
- Il est possible d'ajouter en fin de fichier (append)
- HDFS s'appuie sur le système de gestion de fichier natif. Sous linux sont ext3 ou ext4 le plus souvent.

# HDFS: Quelques commandes

36

❑ Il y a deux possibilités pour manipuler HDFS :

- Soit via l'API Java
- Soit directement depuis un terminal via les commandes

```
$ hdfs dfs <commande hdfs="" />
```

```
$ hadoop fs <commande HDFS>
```

En particulier, les commandes principales sont :

```
hdfs dfs -ls                #listing home dir
hdfs dfs -ls /user          #listing user dir...
hdfs dfs -mv <src><dst>     #Moves files from source to destination.
hdfs dfs -du -h /user       #space used
hdfs dfs -mkdir newdir      #creating dir
hdfs dfs -put myfile.csv .  #storing a file on HDFS
hdfs dfs -get myfile.csv .  #getting a file from HDFS
hdfs dfs -cat myfile.csv .  #edit a file from HDFS
```