Big Data

Principes d'HDFS

Stéphane Vialle **&** **Gianluca Quercini**

1

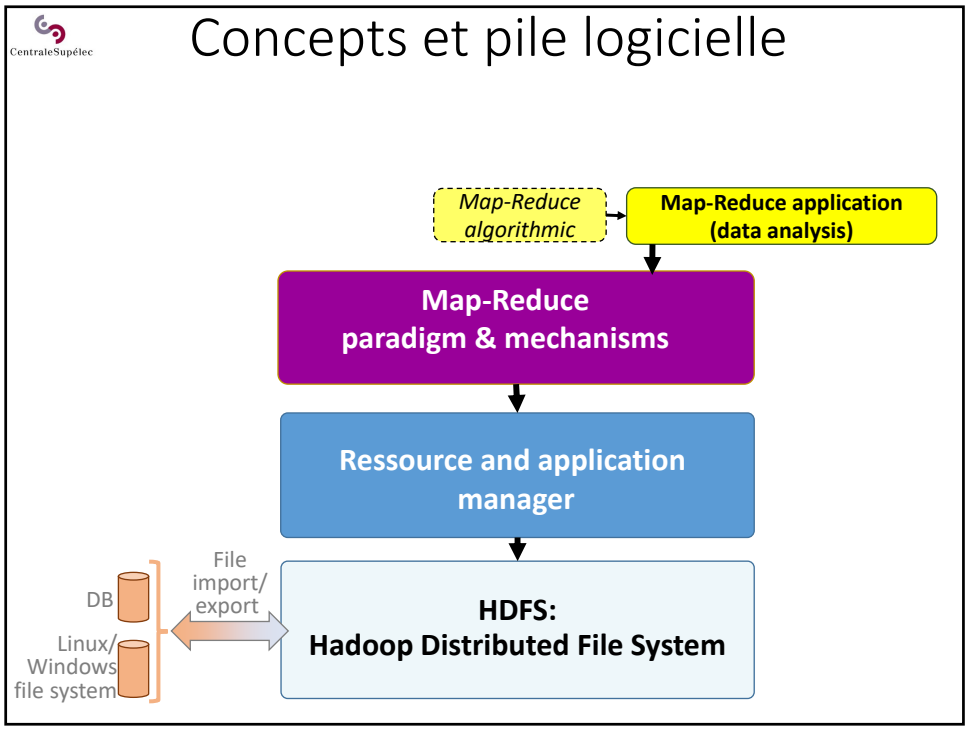
CentraleSupélec

Principes d'HDFS

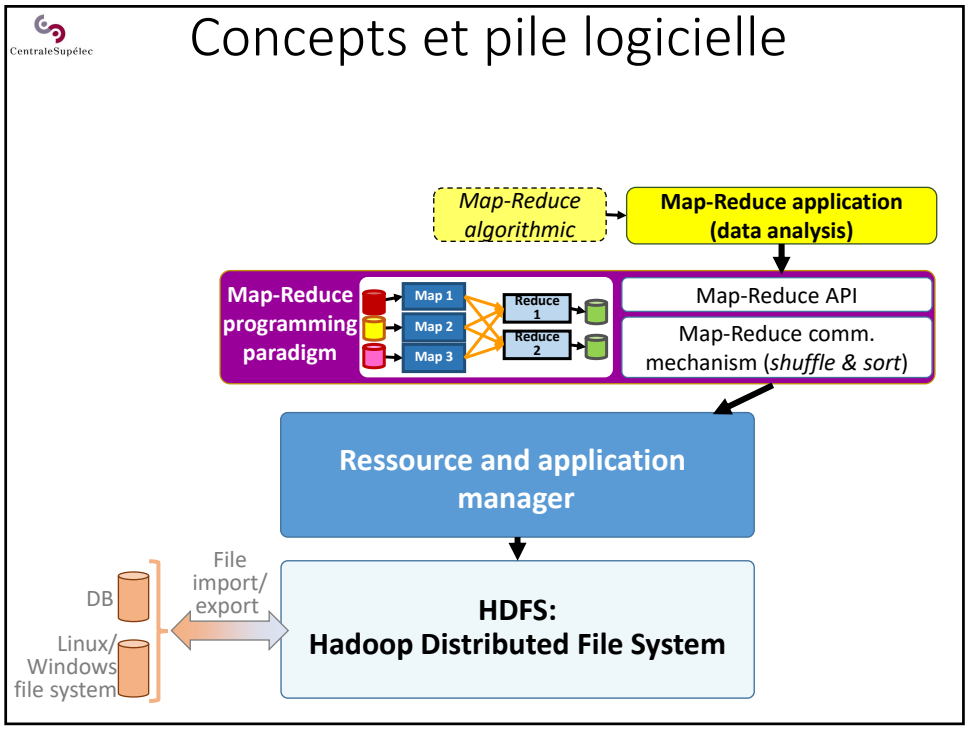
1. Framework d'Hadoop

2. Système de fichiers distribué d'Hadoop : HDFS

2

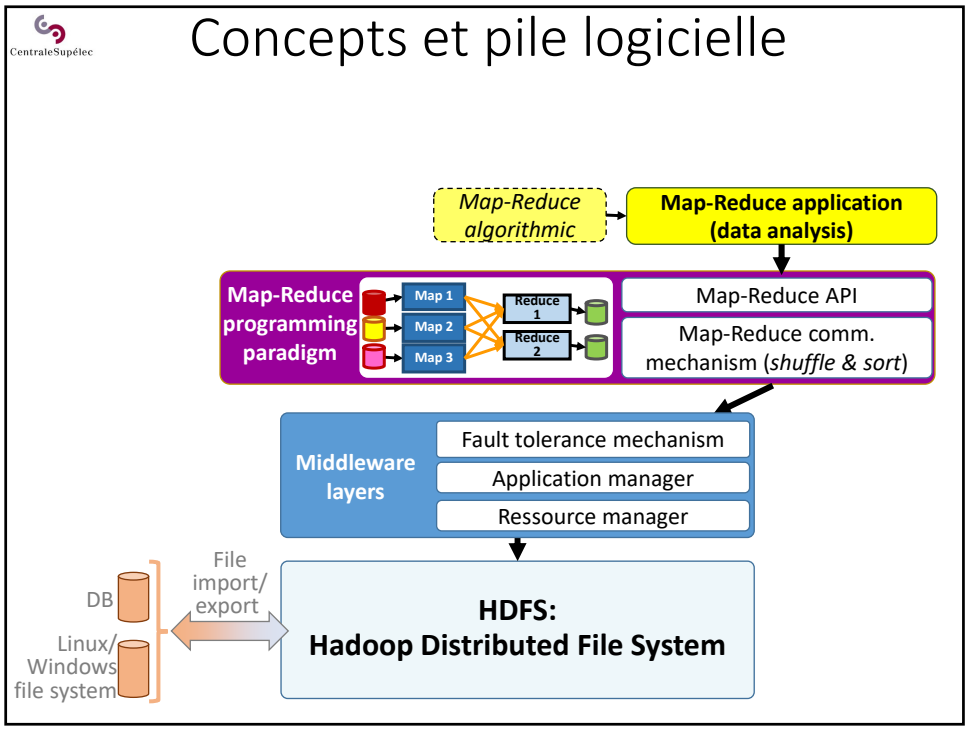

CentraleSupélec
Concepts et pile logicielle
Map-Reduce algorithmic
Map-Reduce application (data analysis)
Map-Reduce paradigm & mechanisms
Ressource and application manager
DB
Linux/ Windows file system
File import/ export
HDFS: Hadoop Distributed File System

3


CentraleSupélec
Concepts et pile logicielle
Map-Reduce algorithmic
Map-Reduce application (data analysis)
Map-Reduce programming paradigm
Map 1
Map 2
Map 3
Reduce 1
Reduce 2
Map-Reduce API
Map-Reduce comm. mechanism (shuffle & sort)
Ressource and application manager
DB
Linux/ Windows file system
File import/ export
HDFS: Hadoop Distributed File System

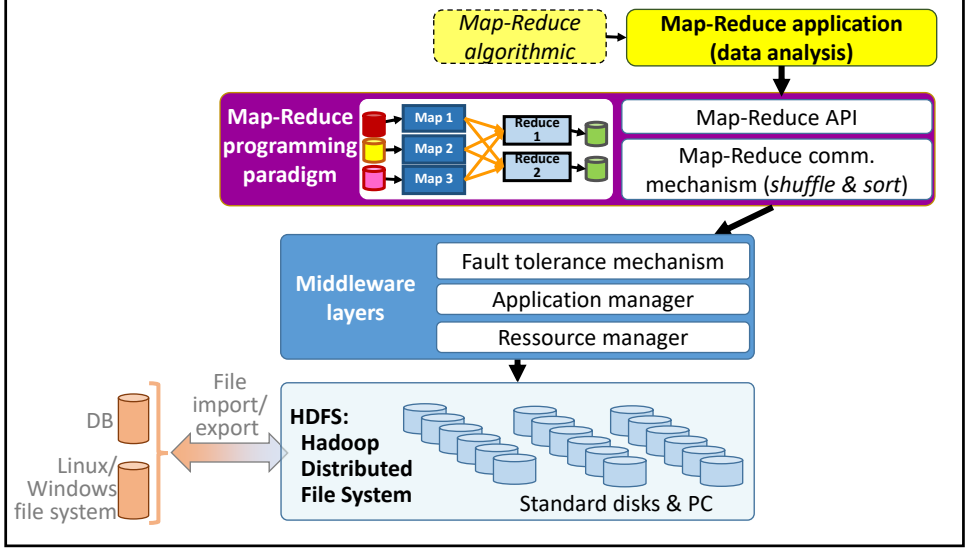
4

Concepts et pile logicielle



5

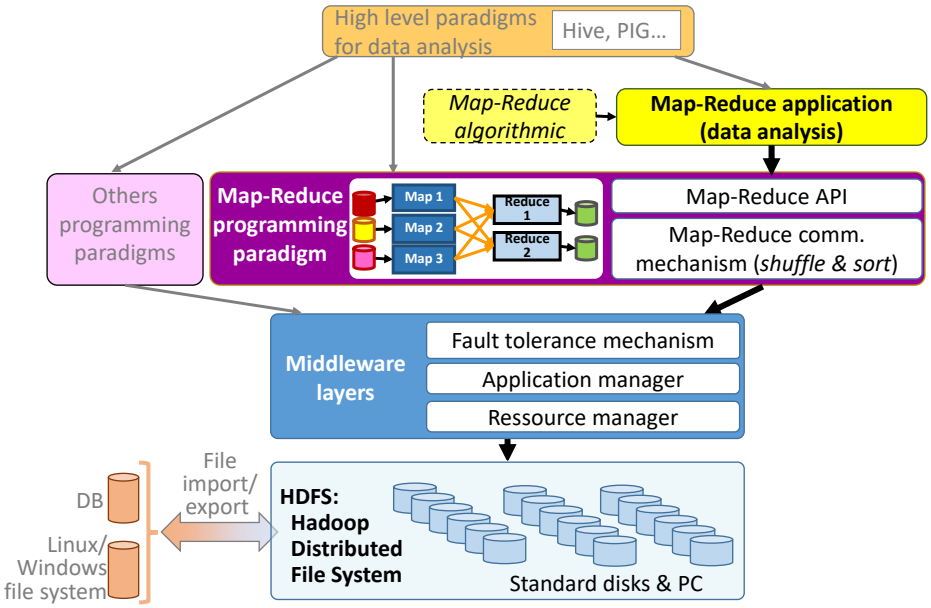
Concepts et pile logicielle



6

CentraleSupélec

Concepts et pile logicielle



7

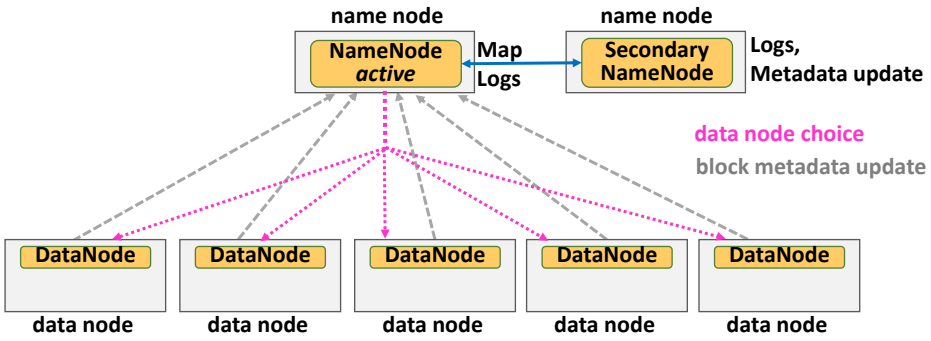
CentraleSupélec

Principes d'HDFS

1. Framework d'Hadoop
2. **HDFS (Hadoop Distributed File System)**
   - **2.1. Stockage des fichiers sous HDFS**
   - 2.2. Mécanisme de haute disponibilité d'HDFS
   - 2.3. Lecture de fichiers sous HDFS
   - 2.4. Ecriture de fichiers sous HDFS

8

Mécanismes d'HDFS

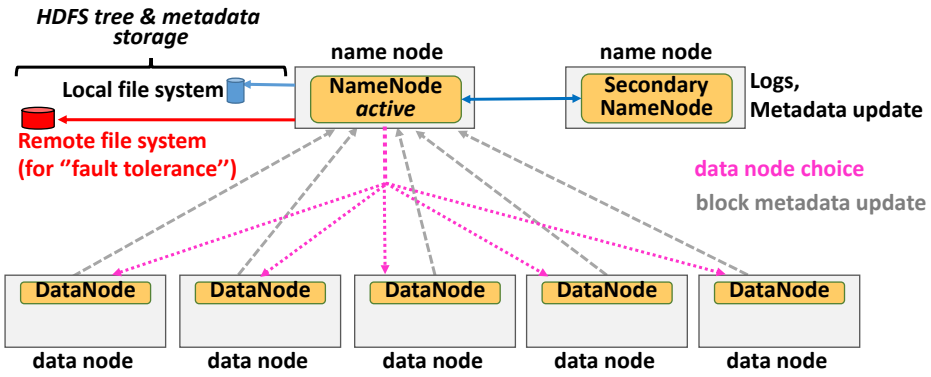


Le *NameNode* conserve la cartographie du HDFS + les évolutions (les « logs »)
→ Permet de savoir où sont les fichiers

Le *secondary NameNode* stocke les logs et met à jour la cartographie (qd bcp de logs)
→ Sur un nœud à part pour pouvoir calculer la mise à jour sans ralentir le système

9

Mécanismes d'HDFS

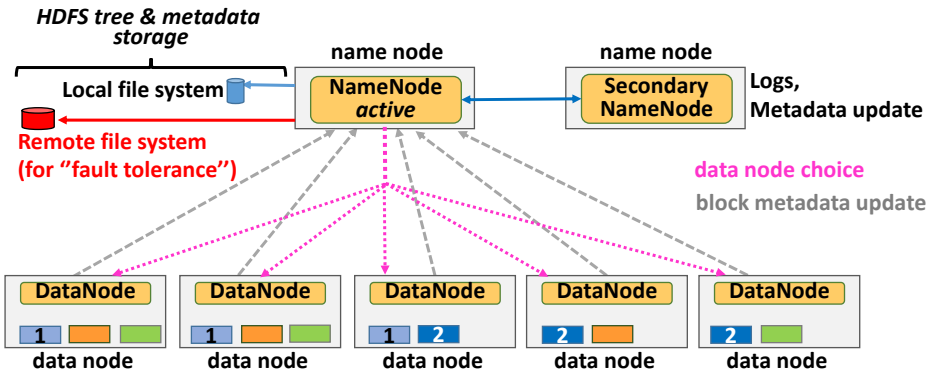


Les metadonnées du HDFS sont stockés sur le file system classique, localement.

Un stockage distant permet de renforcer la tolérance aux pannes
(sans ses metadonnées, le HDFS est inexploitable)

10

Mécanismes d'HDFS

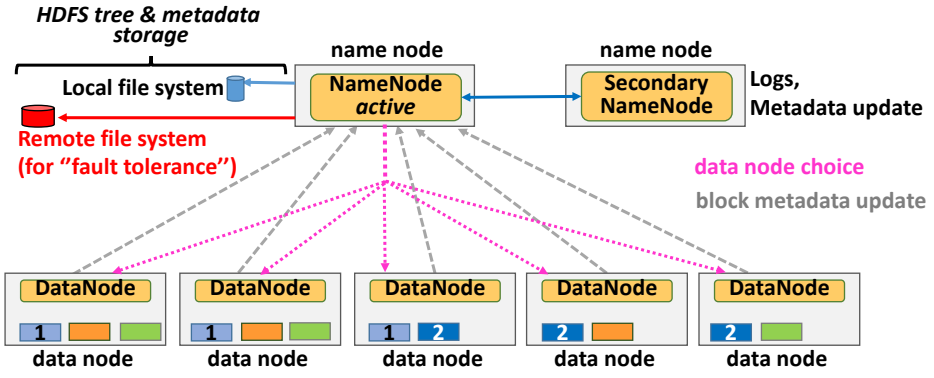


Chaque fichier est découpé en blocs de 64 ou 128 Mo

- **Distribués sur les différents nœuds**
  pour le passage à l'échelle en taille,
  pour la vitesse d'accès
- **Répliqués en n exemplaires (recopiés d'un nœud à un autre)**
  pour la tolérance aux pannes (habituellement n = 3).

11

Mécanismes d'HDFS

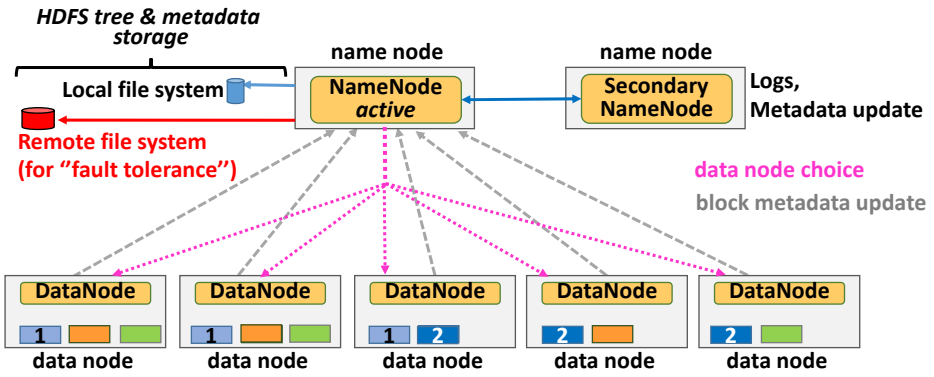


Règles de choix des nœuds :

- Jamais deux réplicats (du même bloc) sur le même nœud
- Des réplicats dans des « raks » différents
  (grands systèmes d'ensembles de raks)

12

Mécanismes d'HDFS



Si un nœud tombe :

- Copie et transfert d'autres réplicats (reconstitution des n réplicats de chaque bloc)
- Envoie des mise à jour des nœuds vers les NameNode

13

Mécanismes d'HDFS

Pourquoi des blocs de 64 Mo ?

Temps de « seek » : temps de positionnement au début du fichier sur le disque (disque standard, rotatif)

Tseek = 10ms

Bande passante disque std : Bw = 100 Mo/s

Tread = Q / Bw

On veut : Tseek < 1% Tread

$\Leftrightarrow 10.10^{-3}$ s < (1/100).(Q/100) s

$\Leftrightarrow$ 100 (Mo) < Q

→ Des blocs de 64 Mo ou 128 Mo permettent de masquer les temps de seek

→ Au-delà : pas plus de gain, mais moins de distribution des fichiers, moins de vitesse de lecture

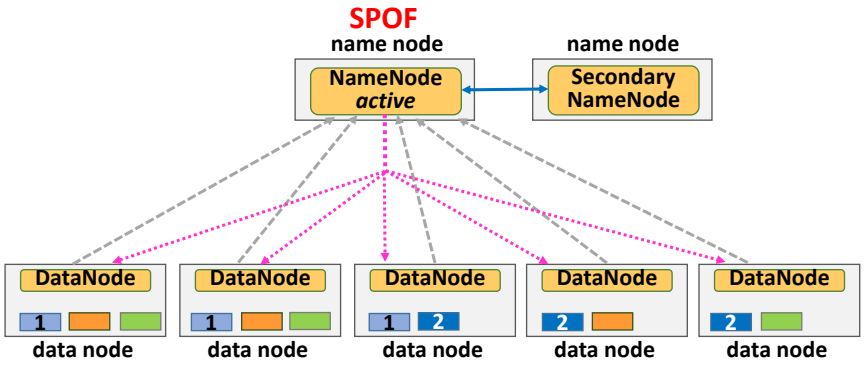
14

Principes d'HDFS

1. Framework d'Hadoop
2. **HDFS (Hadoop Distributed File System)**
   - 2.1. Stockage des fichiers sous HDFS
   - **2.2. Mécanisme de haute disponibilité d'HDFS**
   - 2.3. Lecture de fichiers sous HDFS
   - 2.4. Ecriture de fichiers sous HDFS

15

Mécanismes d'HDFS : « haute disponibilité »

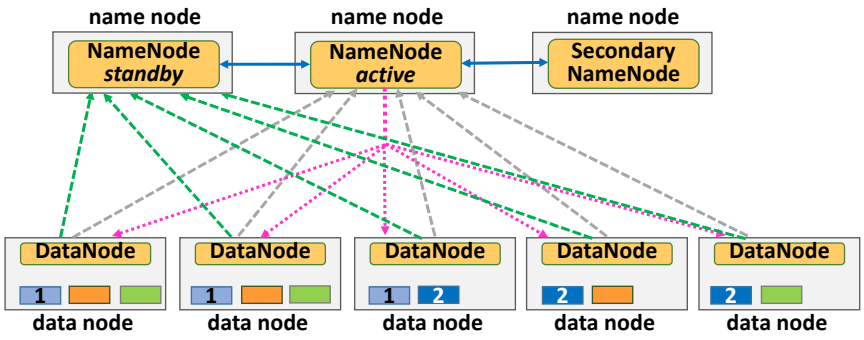


Le NameNode est un SPOF : Single Point Of Failure

→ Si on perd le NameNode alors le File System d'Hadoop ne marche plus !

16

Mécanismes d'HDFS : « haute disponibilité »

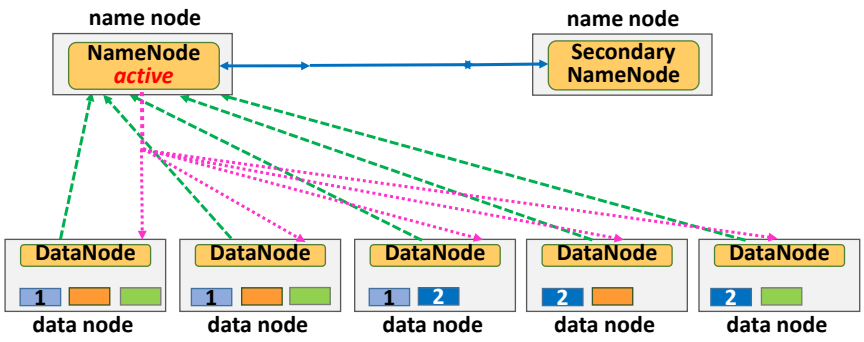


Si on perd le NameNode alors le File System d'Hadoop ne marche plus !

→ On duplique le NameNode

17

Mécanismes d'HDFS : « haute disponibilité »



- Le NameNode en *standy* est passé *actif* « très rapidement »
- Il n'y a plus de SPOF

→ **On « ne sent plus passer la panne » : Haute Disponibilité**

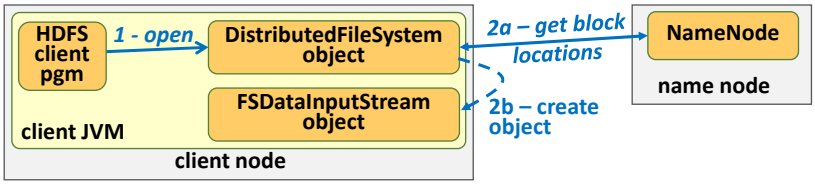
18

Principes d'HDFS

1. Framework d'Hadoop
2. **HDFS (Hadoop Distributed File System)**
   - 2.1. Stockage des fichiers sous HDFS
   - 2.2. Mécanisme de haute disponibilité d'HDFS
   - **2.3. Lecture de fichiers sous HDFS**
   - 2.4. Ecriture de fichiers sous HDFS

19

Mécanismes de lecture d'HDFS



0 – Création d'un objet permettant de s'interfacer à HDFS (un stub/proxy d'HDFS)

1 – Demande d'ouverture d'un fichier HDFS en lecture (« open »)

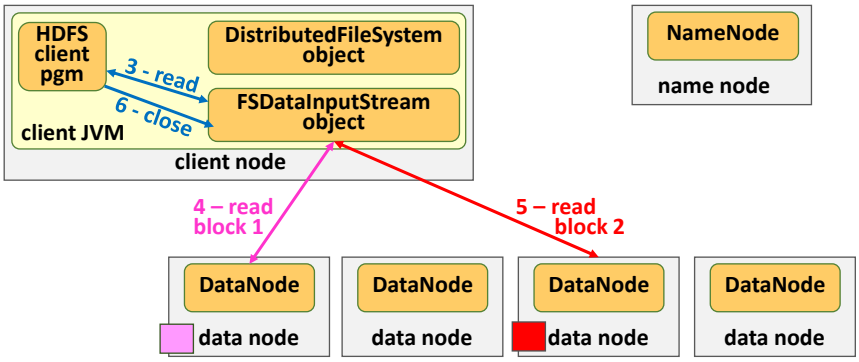
2 – Demande de localisation du fichier

2a - Le proxy interroge le NameNode : pour savoir quels blocs lire et où les lire
Le NameNode répond au proxy

2b - Le proxy crée et retourne un objet lecteur spécialisé sur le fichier ciblé

20

Mécanismes de lecture d'HDFS



3 – Le client exploite le lecteur du fichier

4 – Le lecteur accède au data node du premier bloc et récupère ce bloc

5 – Le lecteur accède au data node du second bloc et récupère ce bloc

6 – Le client referme le fichier en s'adressant au lecteur du fichier

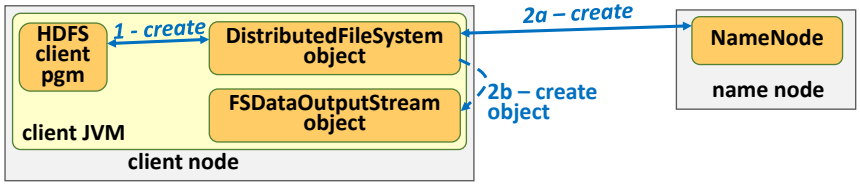
21

Principes d'HDFS

1. Framework d'Hadoop
2. **HDFS (Hadoop Distributed File System)**
   - 2.1. Stockage des fichiers sous HDFS
   - 2.2. Mécanisme de haute disponibilité d'HDFS
   - 2.3. Lecture de fichiers sous HDFS
   - **2.4. Ecriture de fichiers sous HDFS**

22

Mécanisme d'écriture d'HDFS



0 – Création d'un objet permettant de s'interfacer à HDFS (un stub/proxy d'HDFS)

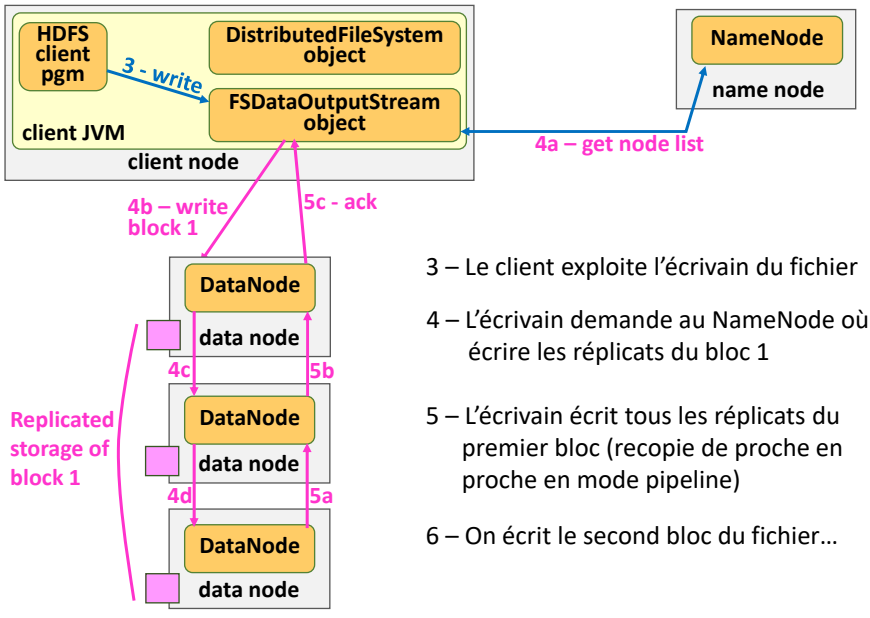
1 – Demande d'ouverture d'un fichier HDFS en écriture (« create »)

2 – Demande de localisation des nœuds d'accueil des futurs blocs du fichier
2a - Le proxy interroge le NameNode : pour savoir où écrire les blocs
Le NameNode répond au proxy

2b - Le proxy crée et retourne un objet écrivain spécialisé sur le fichier ciblé

23

Mécanisme d'écriture d'HDFS



3 – Le client exploite l'écrivain du fichier

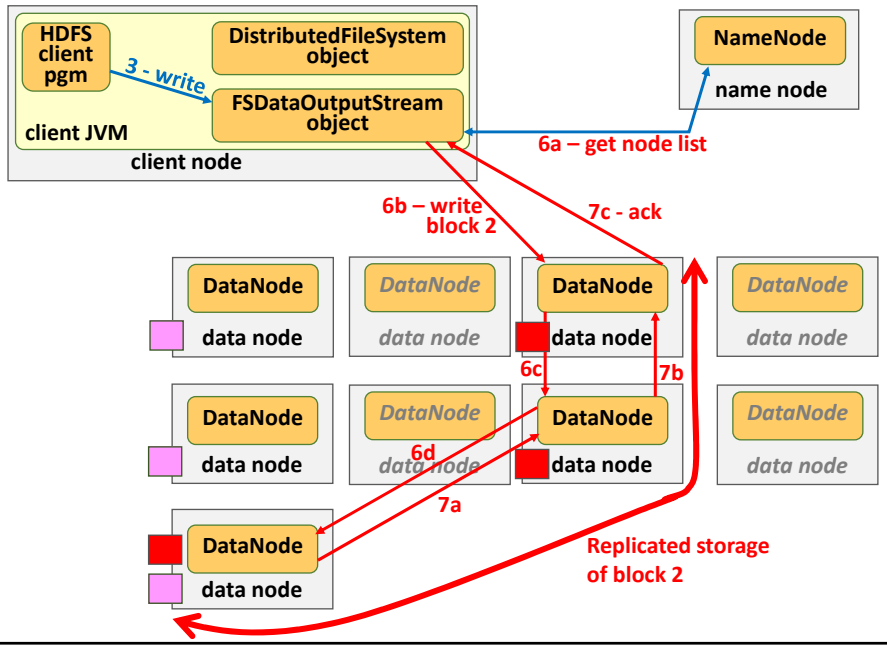
4 – L'écrivain demande au NameNode où écrire les réplicats du bloc 1

5 – L'écrivain écrit tous les réplicats du premier bloc (recopie de proche en proche en mode pipeline)

6 – On écrit le second bloc du fichier…

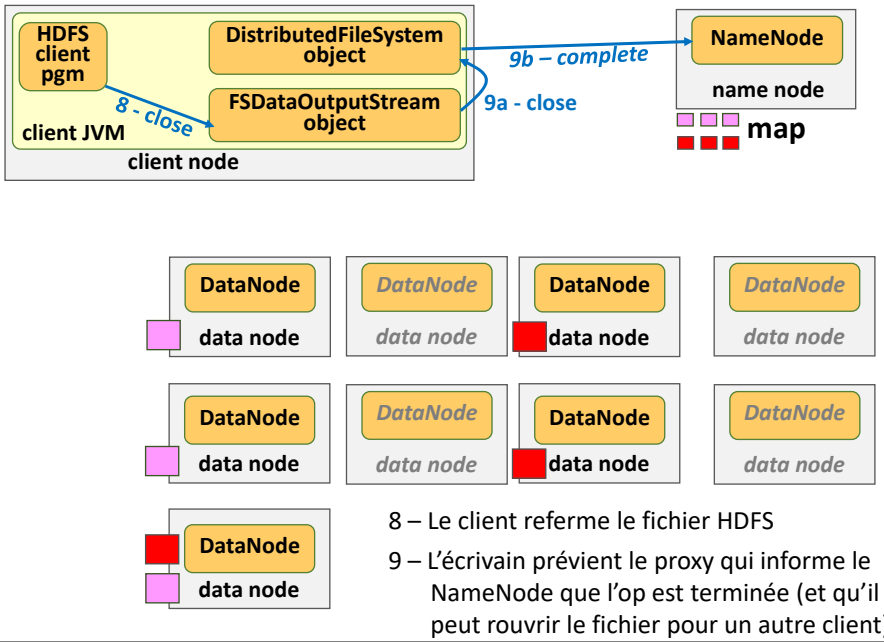
24

Mécanisme d'écriture d'HDFS



25

Mécanisme d'écriture d'HDFS



8 – Le client referme le fichier HDFS

9 – L'écrivain prévient le proxy qui informe le NameNode que l'op est terminée (et qu'il peut rouvrir le fichier pour un autre client)

26

CentraleSupélec

QUIZ

Q1: a user connects his client program, running on his laptop, to a 100-node Hadoop cluster, and submits Map-Reduce queries, to compute the histogram of the age of the French (with one-year increments)

→ Technically, can he download the results to his laptop?

→ Technically, can he upload new input data to the HDFS of the 100-node cluster?

→ Technically, can he download the input data to his laptop and then load it into a second Hadoop cluster?

→ Is it possible to copy data from the HDFS of a first Hadoop cluster directly to the HDFS of a second?

27

CentraleSupélec

QUIZ

Q2: a failure occurs on a Hadoop data node used during the execution of a Map-Reduce program (the node disappears)

→ Does the user have to resubmit the Map-Reduce request?

→ Does the user get the result later when a failure occurs?

Q3: to improve fault tolerance, you can install HDFS on top of a RAID-enabled storage array (*Redundant Array of Independent Disks*)

→ Do you think this is a logical approach?

28

CentraleSupélec

Principes d'HDFS

29