






 CentraleSupélec  


Big Data : Informatique pour les données et calculs massifs

5 – Technologies d'Hadoop

Stéphane Vialle

 université  PARIS-SACLAY  ÉCOLE DOCTORALE Sciences et technologies de l'information et de la communication (STIC)  RISEGrid  lorraine  Grand Est ALSACE CHAMPAGNE-ARDENNE LORRAINE

Stephane.Vialle@centralesupelec.fr
<http://www.metz.supelec.fr/~vialle>

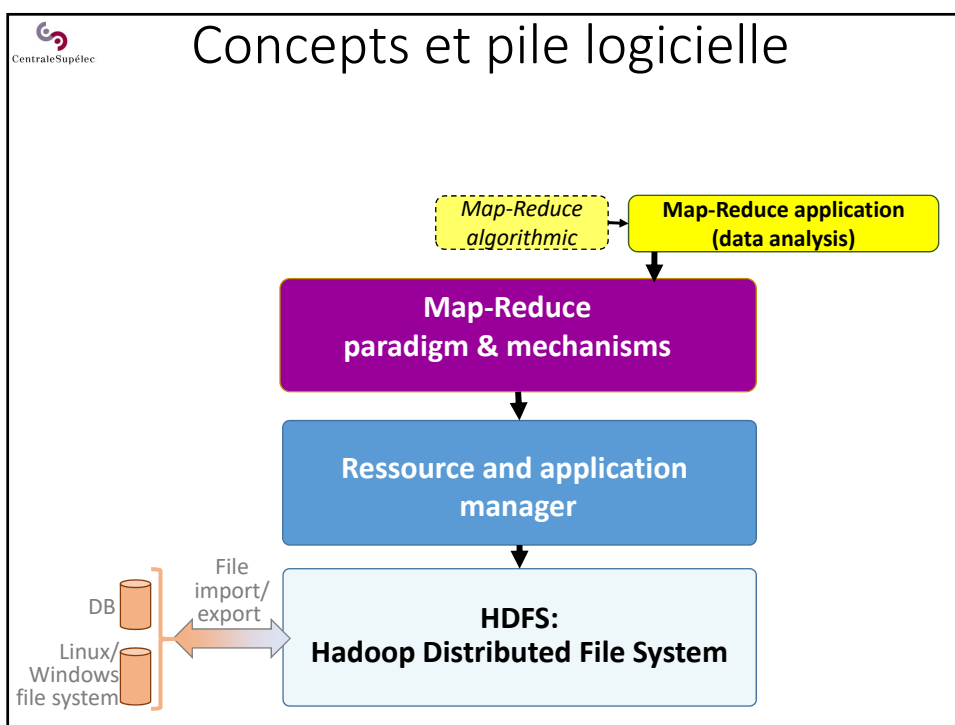
 CentraleSupélec

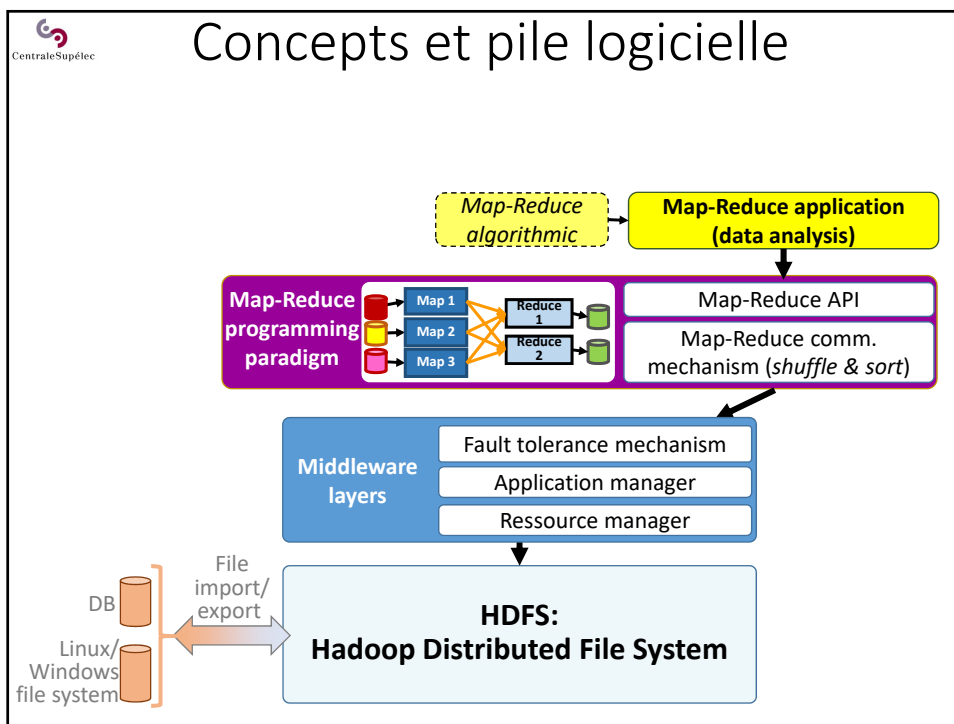
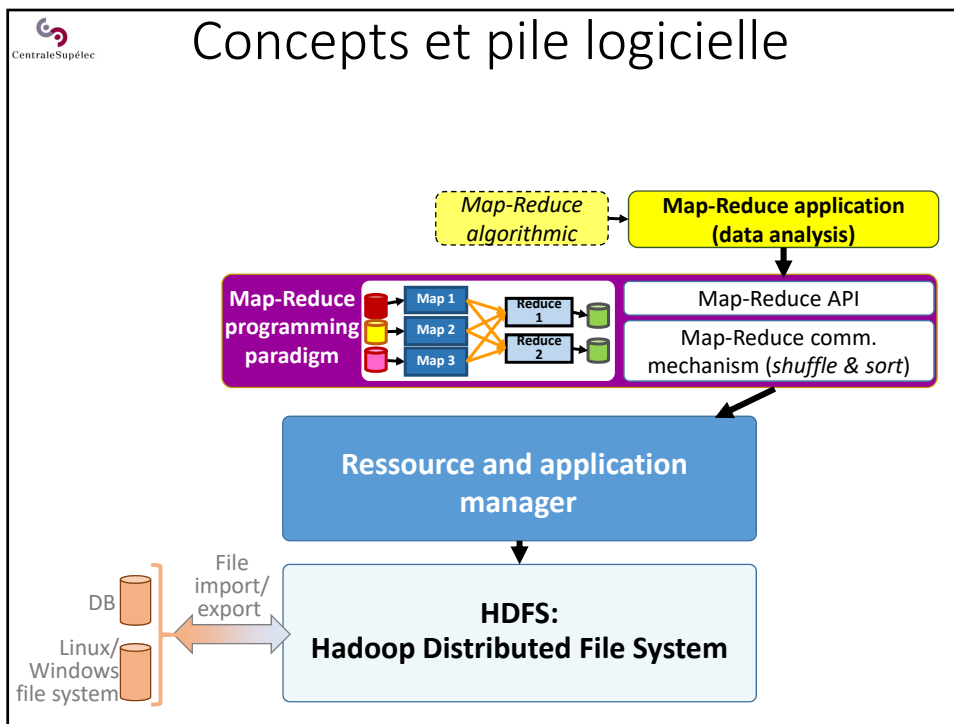
Technologies d'Hadoop

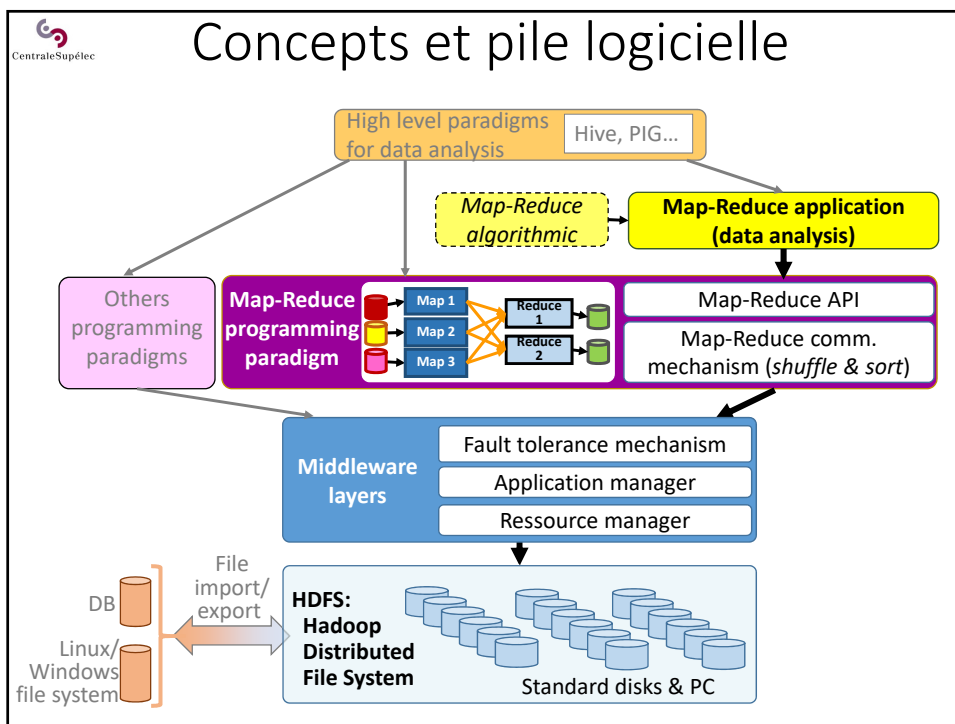
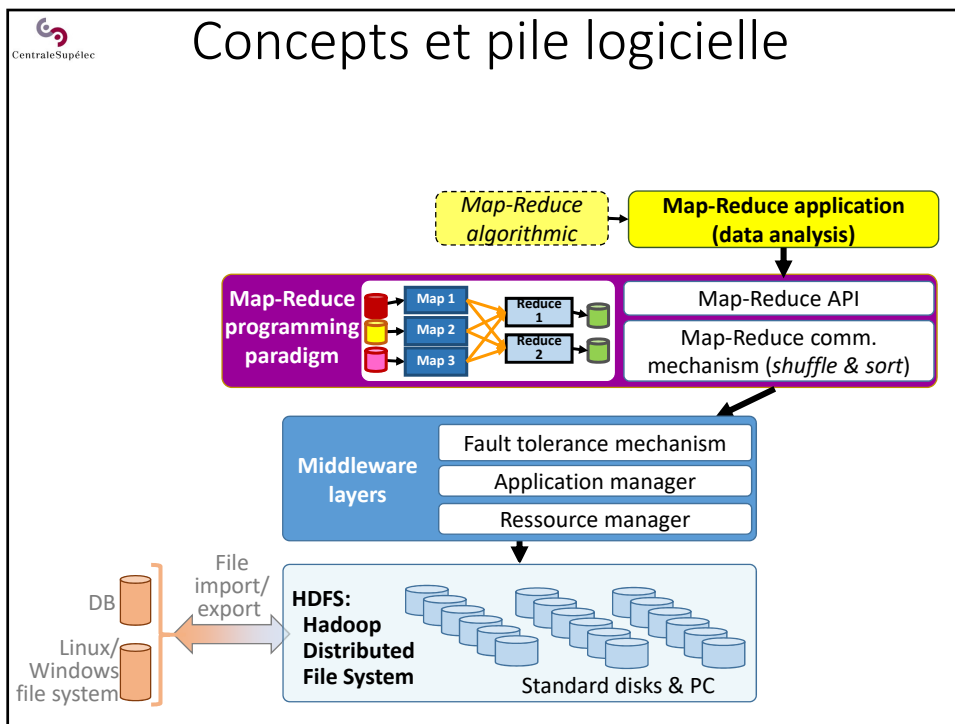
1. Principaux composants d'Hadoop
2. Principes du Map-Reduce d'Hadoop
3. Système de fichiers distribué d'Hadoop : HDFS
4. Allocation et gestion de ressources d'Hadoop ...
...pour un Map-Reduce


CentraleSupélec

1 - Principaux composants d'Hadoop









Concepts de « localité des données et des traitements »


C'est toujours l'accès aux données qui coute cher, pas le calcul
lui-même une fois les données dans le cœur de calcul

HPC :

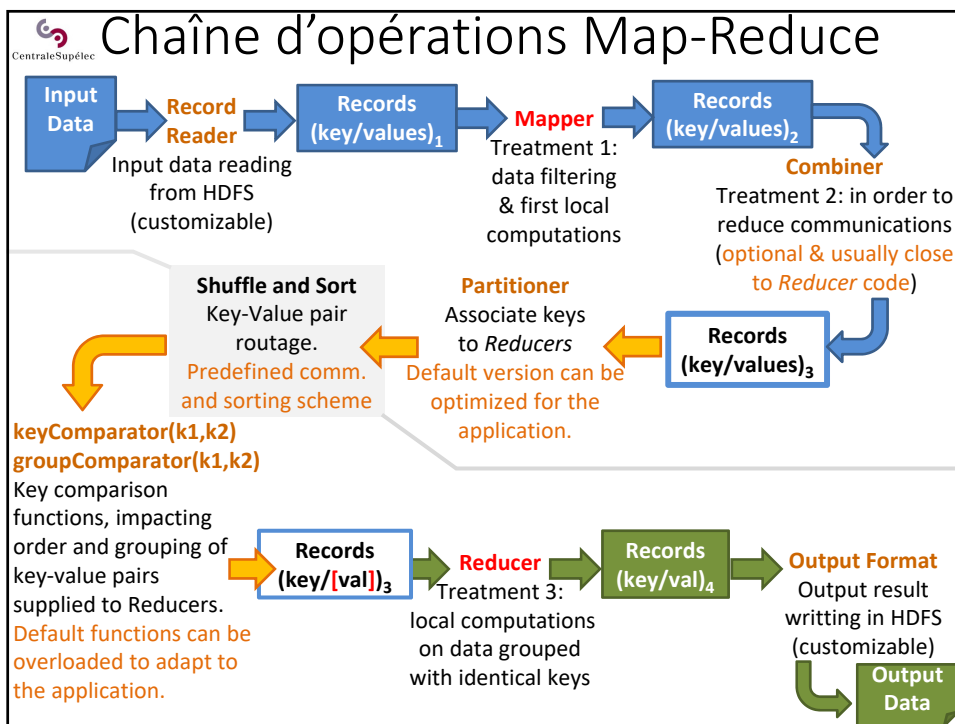
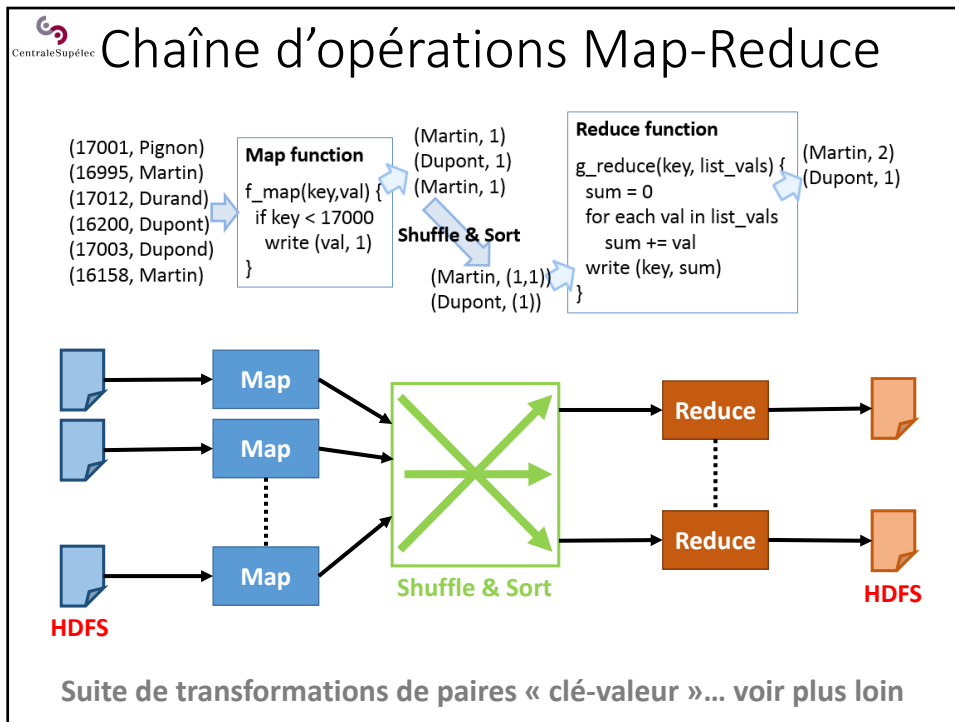
- Charger les données en RAM (utiliser assez de nœuds de calcul)
- Calculer tout ce qui est possible avec des données locales stockées dans la mémoire cache (calcul par bloc/tuile)
- Eviter les défauts de cache

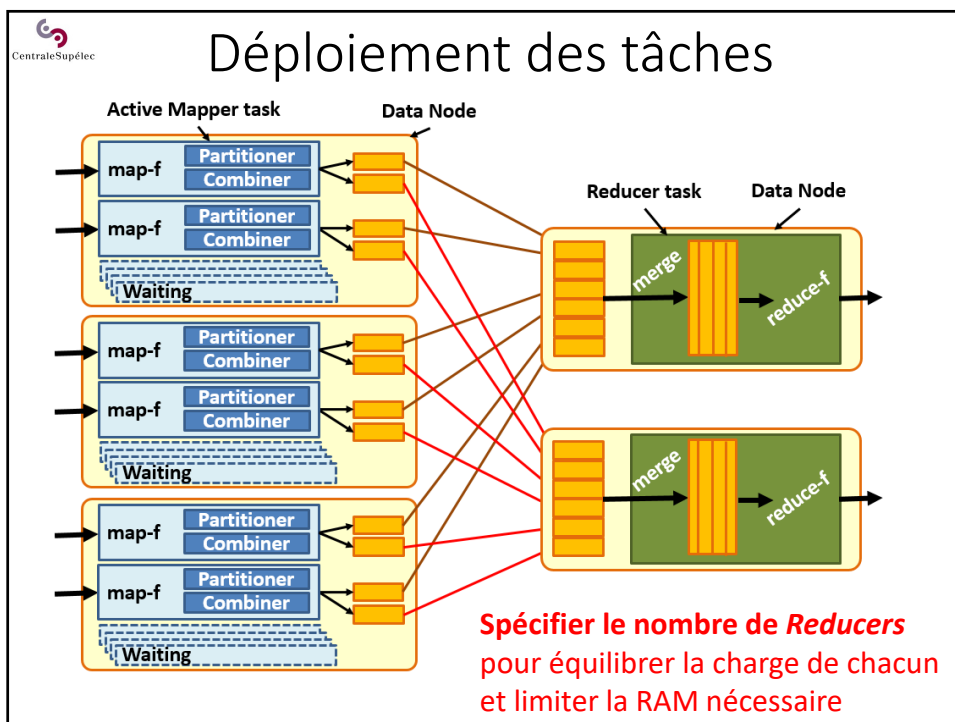
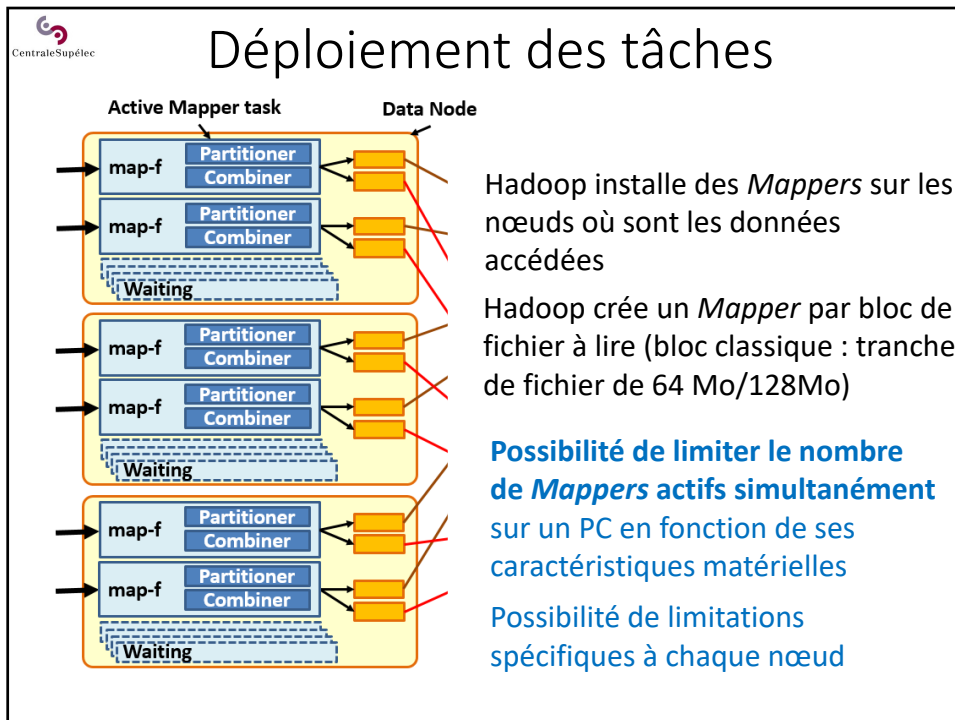
Big Data :

- **Amener les codes de traitements aux données**
→ **Transformer momentanément les nœuds de stockage en nœuds de traitement**
- Eviter de déplacer des données (très volumineuses)



2 - Principes du Map-Reduce d'Hadoop

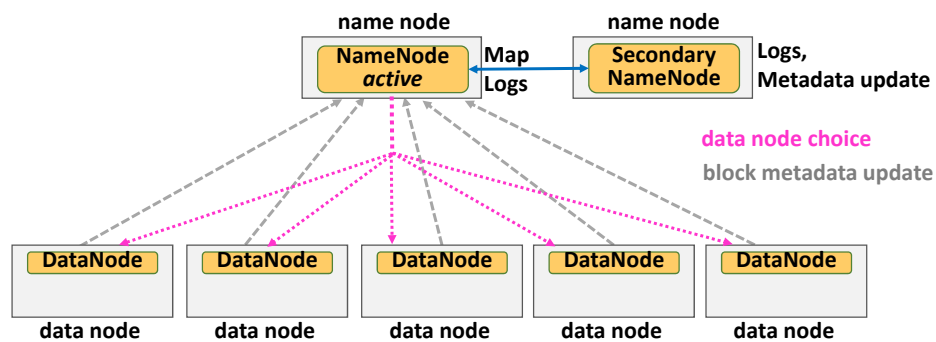




3 - Système de fichiers distribué d'Hadoop : HDFS

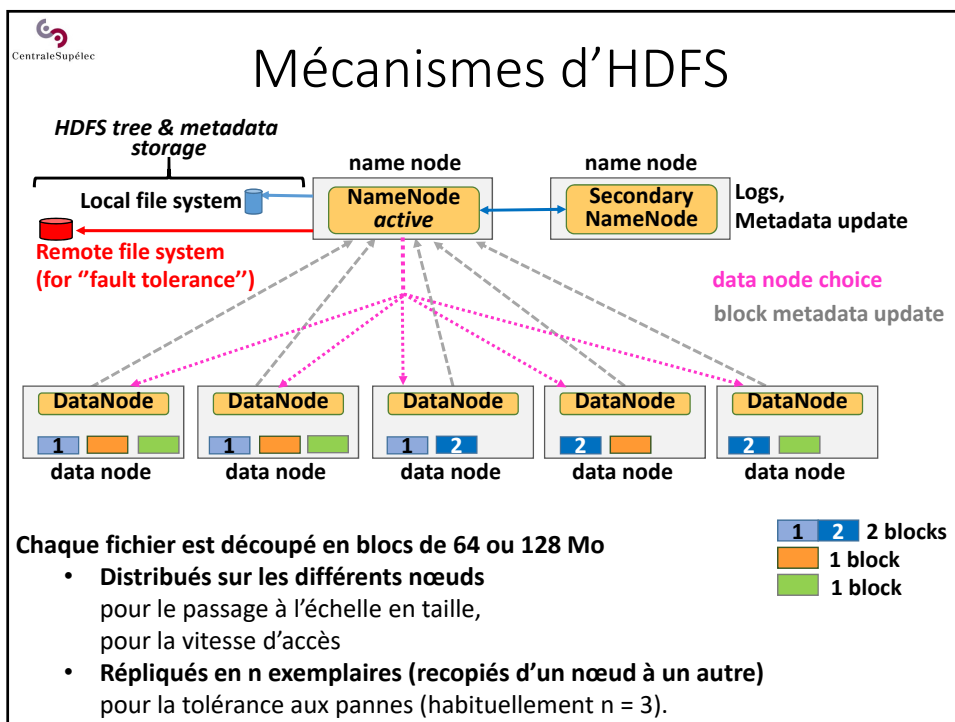
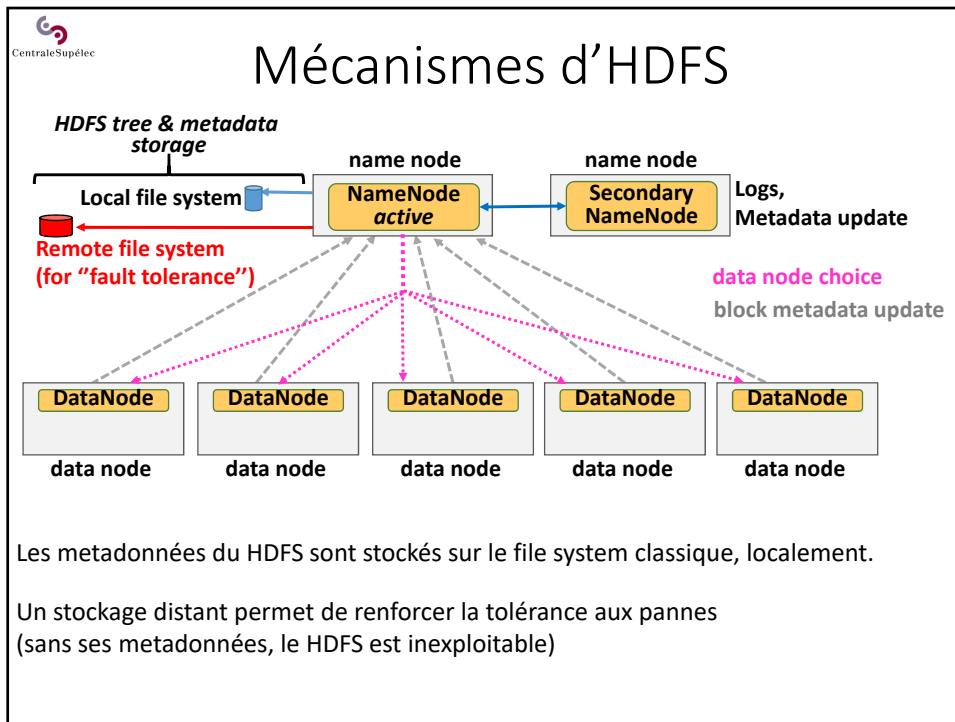
3.1 - Distribution et réplcation des fichiers sous HDFS

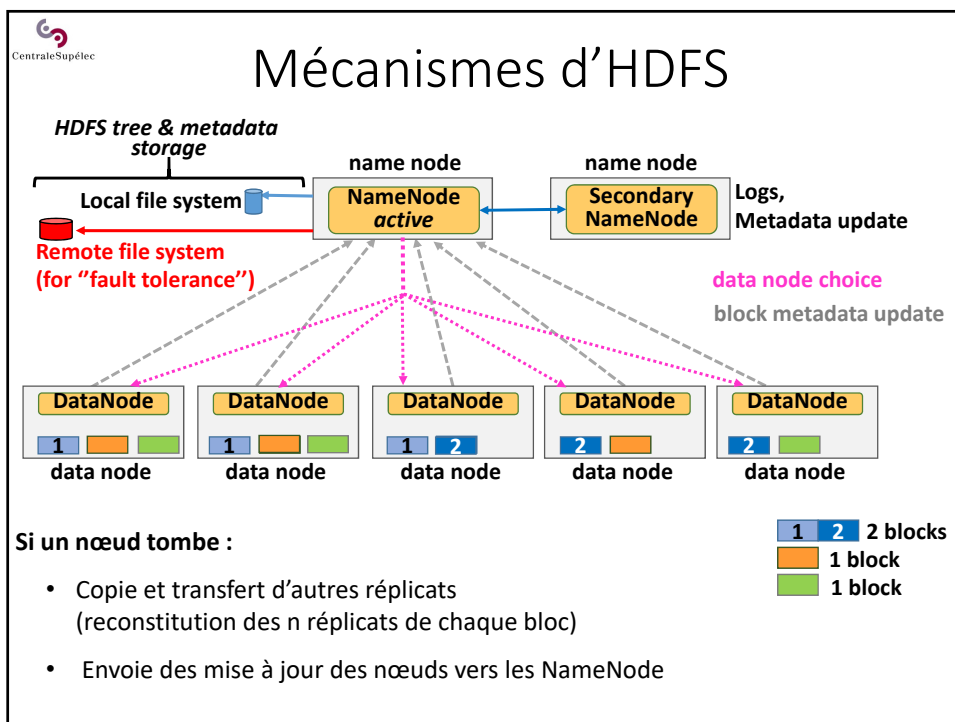
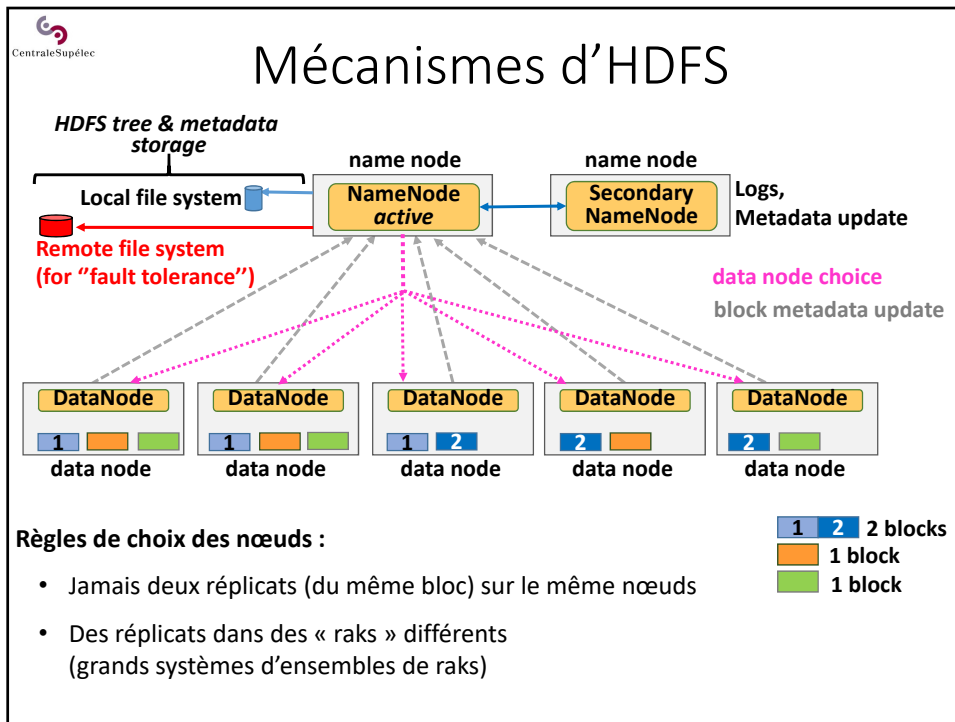
Mécanismes d'HDFS



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





CentraleSupélec

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)
 $T_{seek} = 10\text{ms}$

Bande passante disque std : $Bw = 100\text{ Mo/s}$
 $T_{read} = Q / Bw$

On veut : $T_{seek} < 1\% T_{read}$
 $\Leftrightarrow 10 \cdot 10^{-3}\text{ s} < (1/100) \cdot (Q/100)\text{ s}$
 $\Leftrightarrow 100\text{ (Mo)} < Q$

→ 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

CentraleSupélec

Mécanismes d'HDFS : « haute disponibilité »

SPOF
 name node
 NameNode active
 name node
 Secondary NameNode

DataNode
 1
 data node

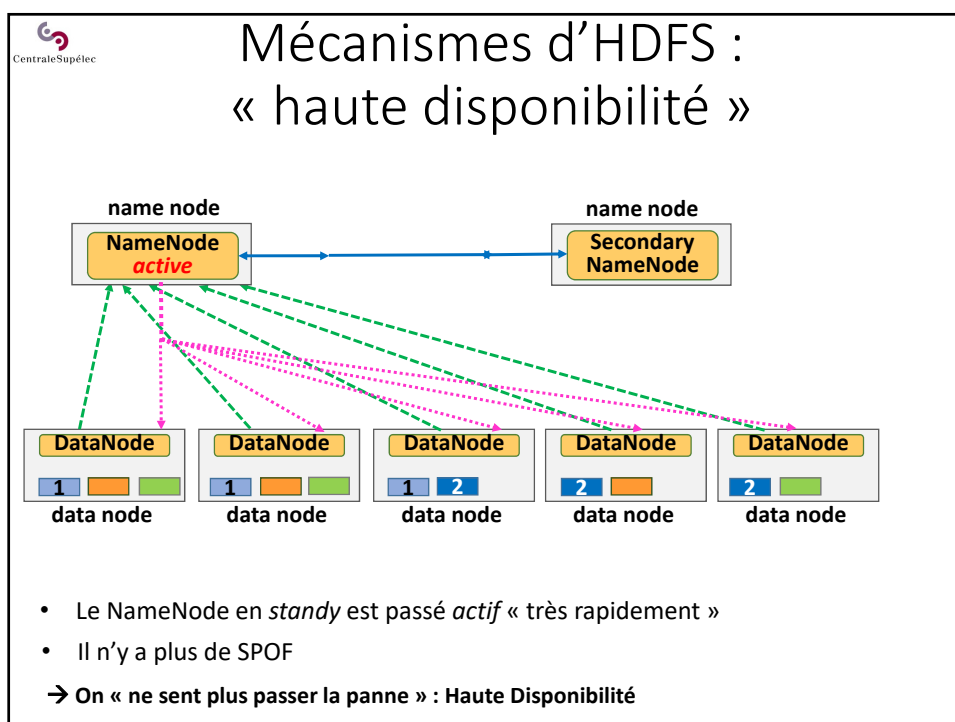
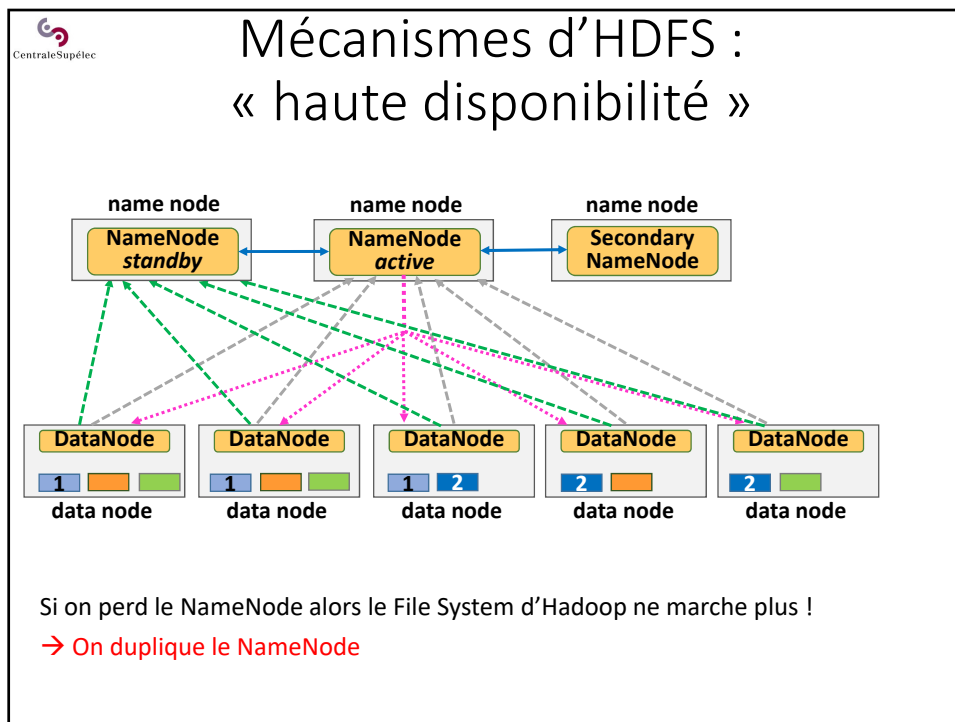
DataNode
 1
 data node

DataNode
 1 2
 data node

DataNode
 2
 data node

DataNode
 2
 data node

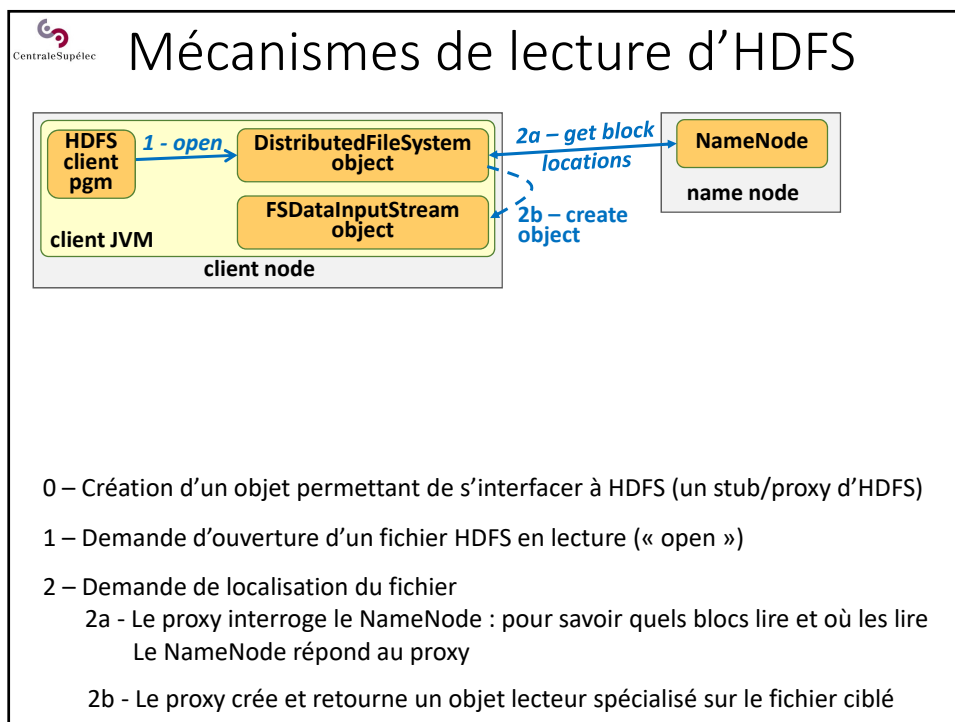
Le NameNode est un SPOF : Single Point Of Failure
 → Si on perd le NameNode alors le File System d'Hadoop ne marche plus !

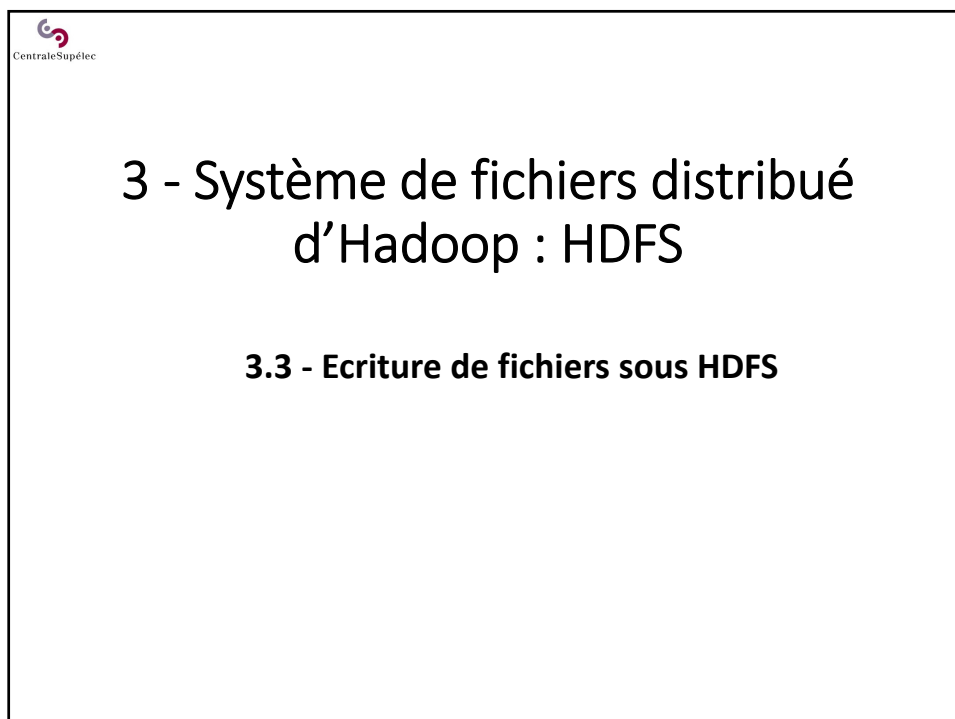
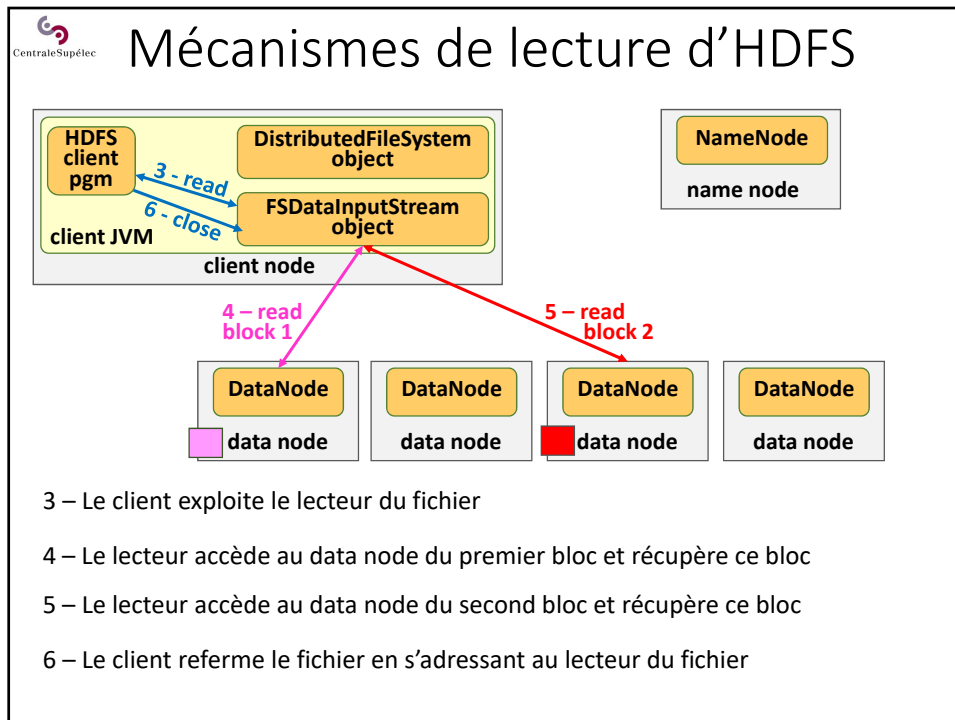


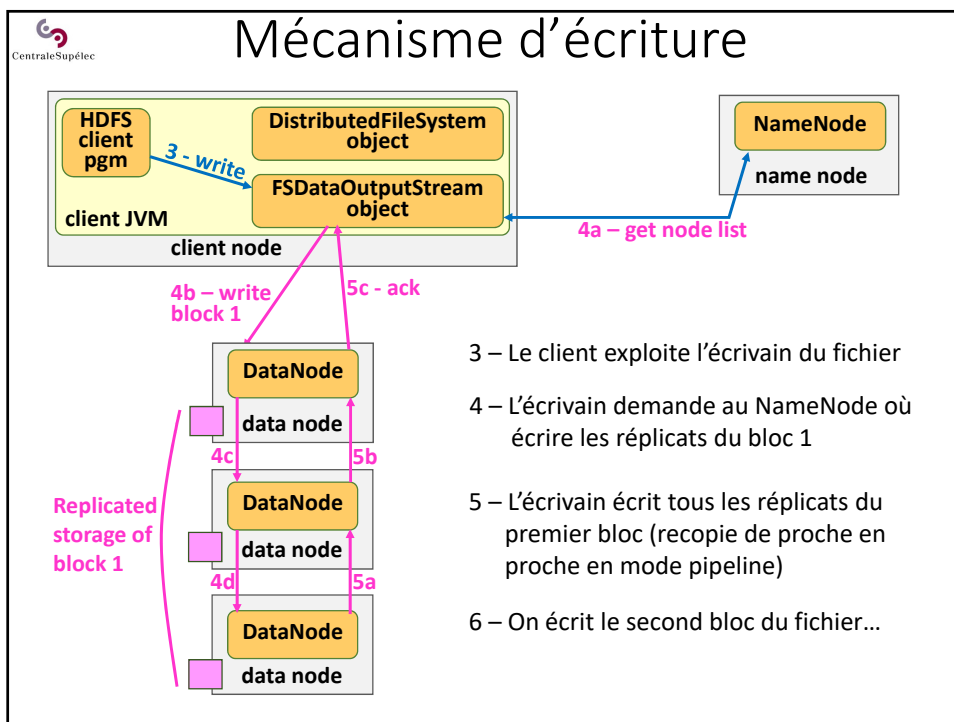
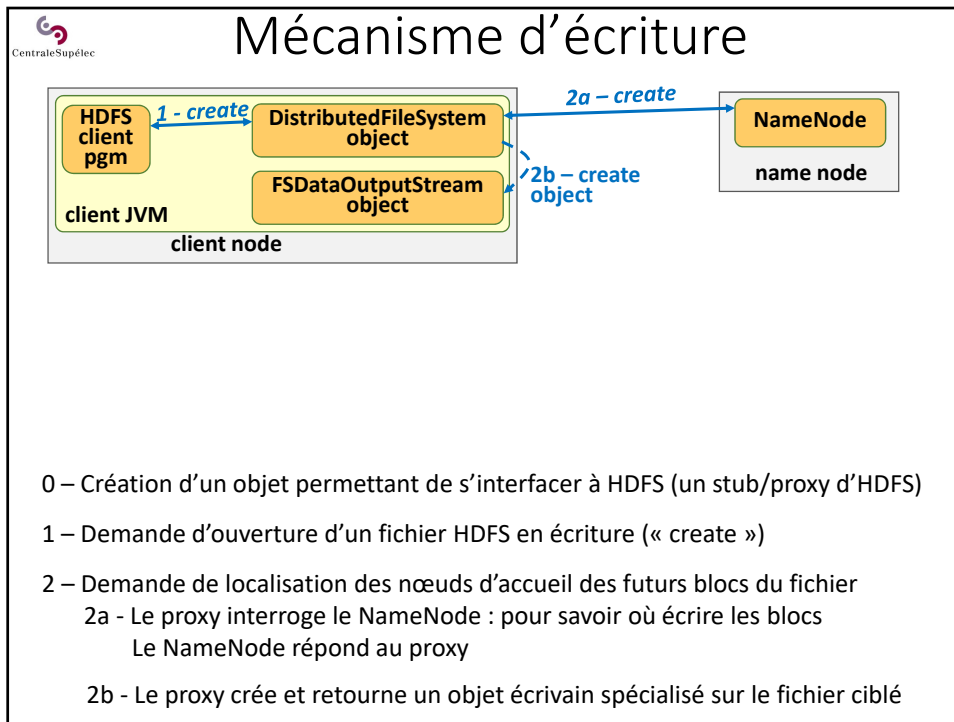
CentraleSupélec

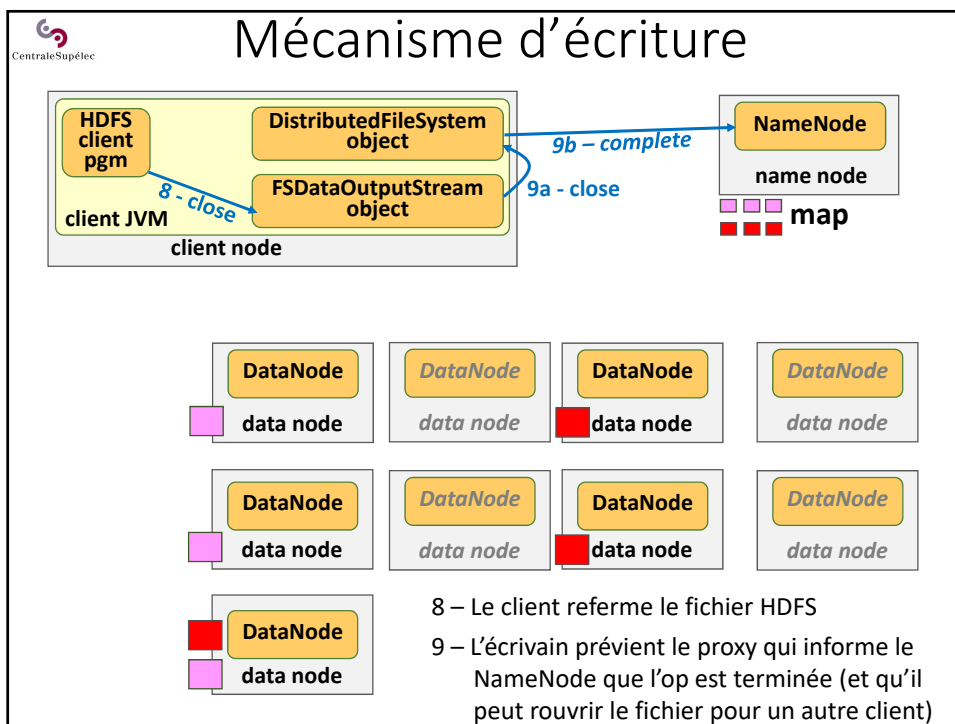
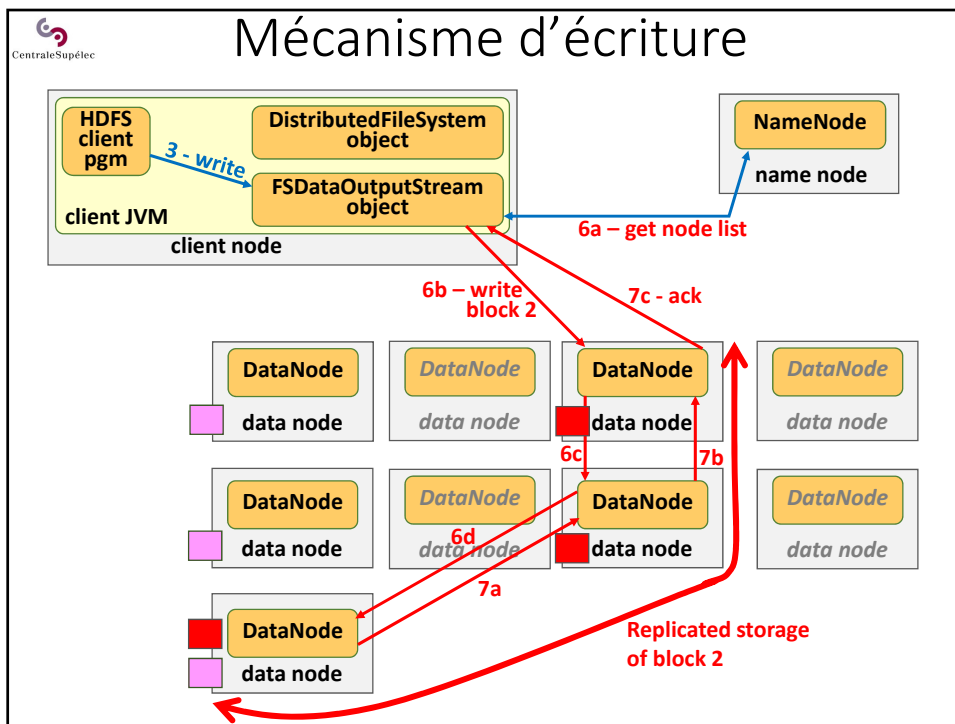
3 - Système de fichiers distribué d'Hadoop : HDFS

3.2 - Lecture de fichiers sous HDFS











3 - Système de fichiers distribué d'Hadoop : HDFS

3.4 - Bilan des accès aux fichiers sous HDFS



Bilan des accès aux fichiers HDFS

Le client crée un objet « proxy » (ou « stub ») local qui fait toute l'interface avec l'architecture HDFS (NameNode et DataNodes) :

- SIMPLE
- **Pb 1 : Tous les accès (R/W) sont séquentiels** : bloc après bloc, et séquentiel au sein d'un bloc !
- **Pb 2 : Le client n'aura pas assez de RAM ou même de disque** pour rapatrier en local un gros fichier HDFS !

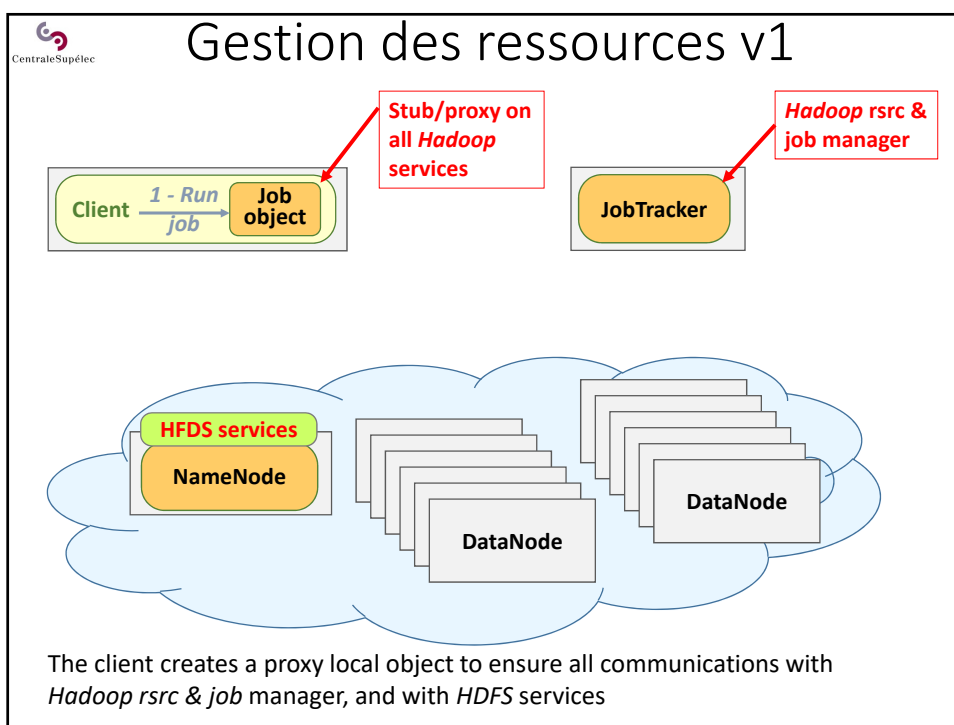
→ OK pour importer/exporter des données vers d'autres File Systems ou BdD (sera long mais fonctionnera)

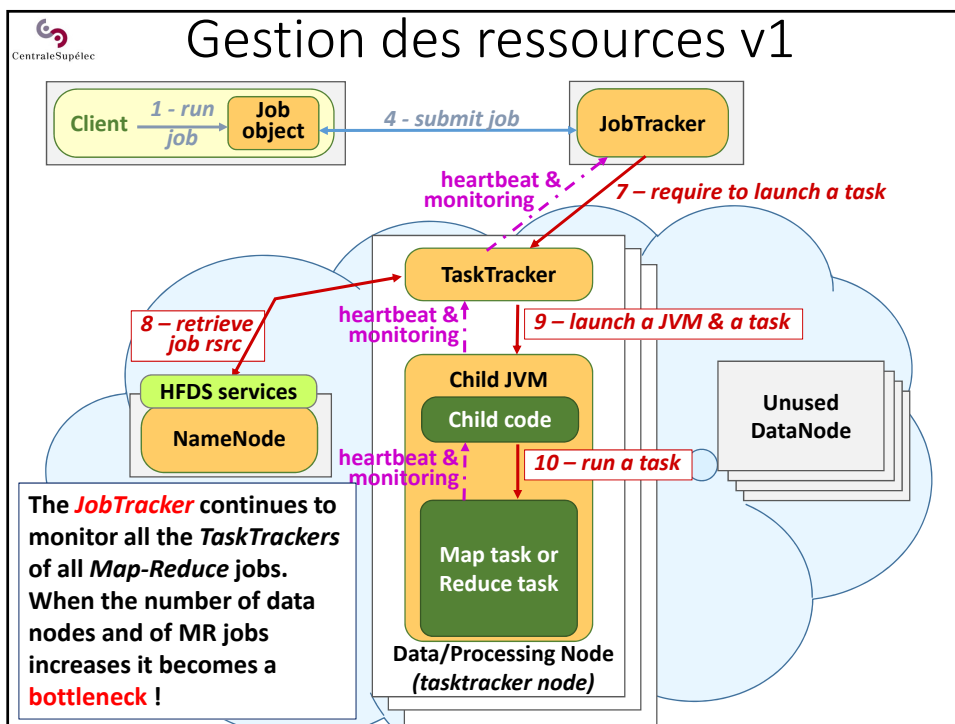
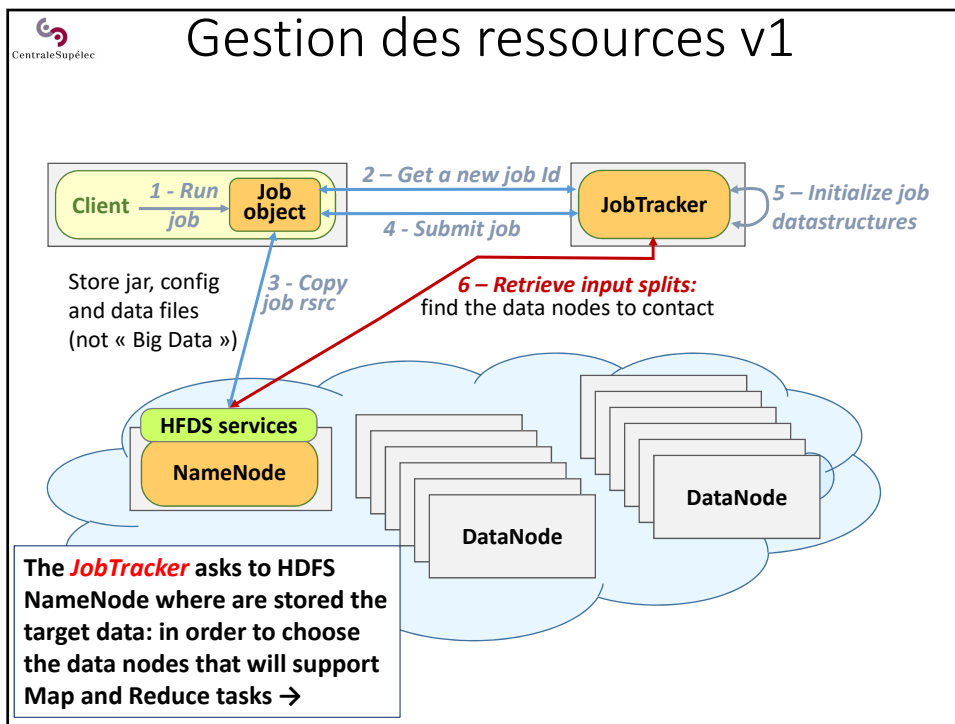
→ **PAS la bonne stratégie pour traiter des fichiers HDFS (des Big Data) : il manque la localisation Data/Calculs**

CentraleSupélec

4 - Allocation et gestion de ressources d'Hadoop... ... pour un Map-Reduce

Version 1

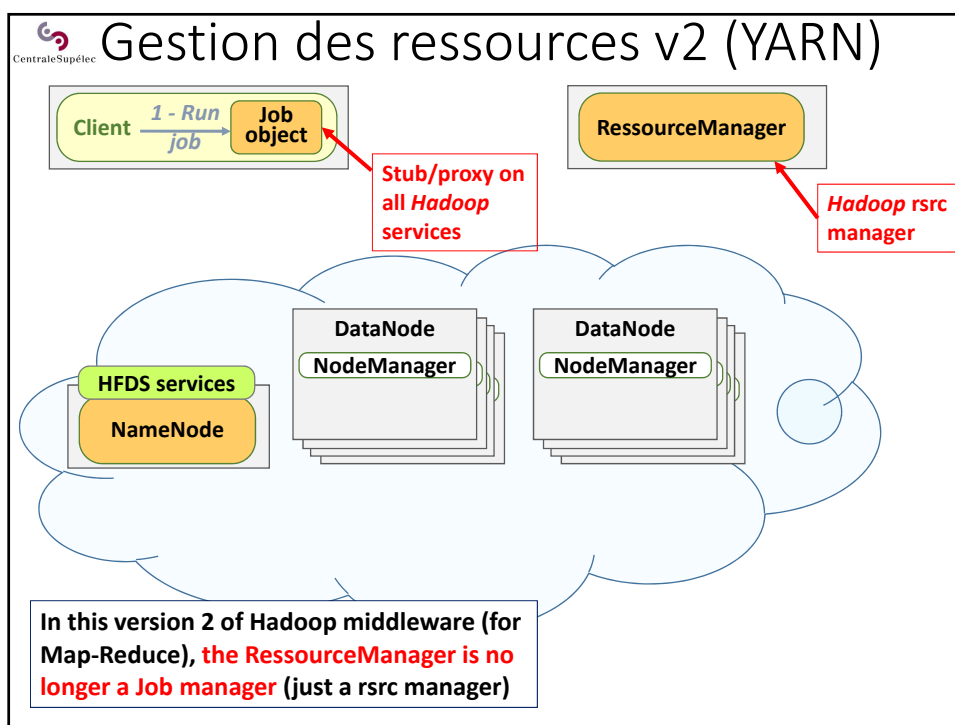


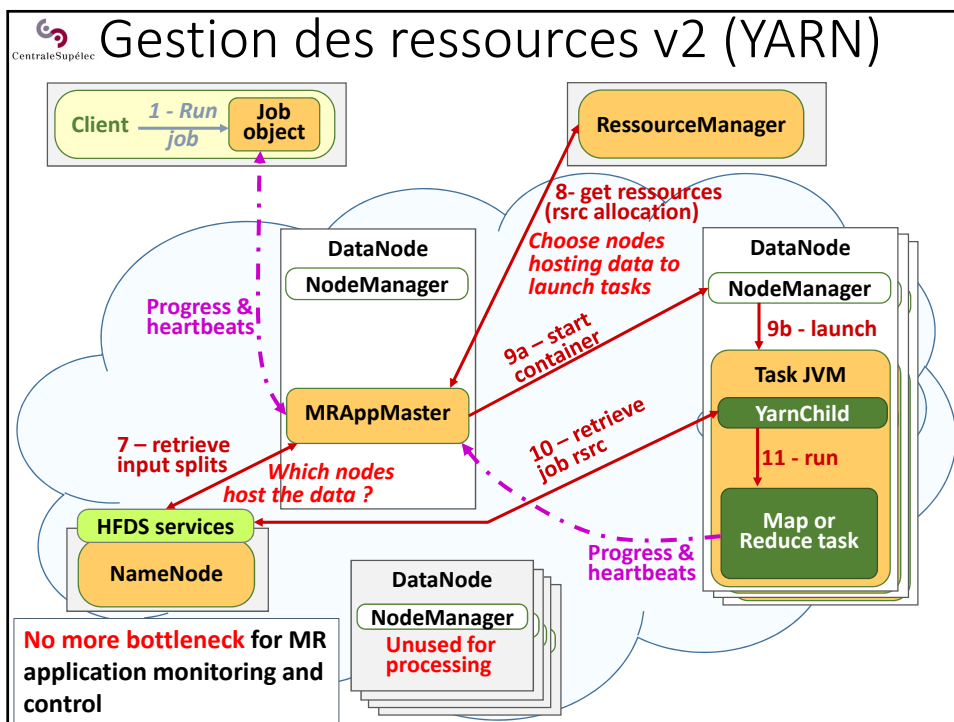
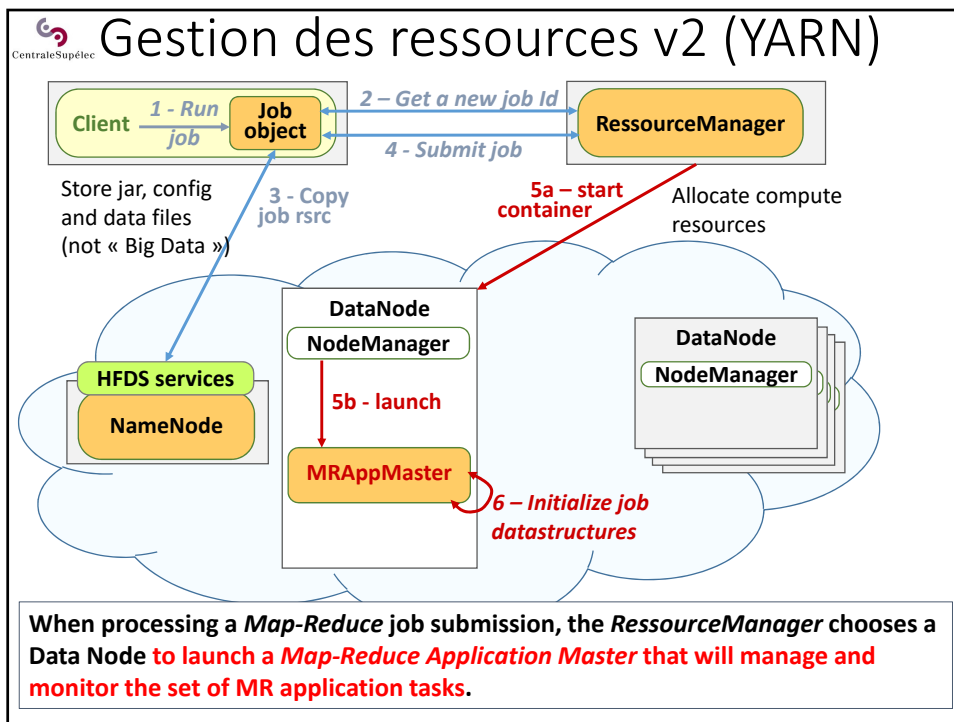


CentraleSupélec

4 - Allocation et gestion de ressources d'Hadoop... ... pour un Map-Reduce

Version 2 (YARN) : passage à l'échelle amélioré





Exécution spéculative

Hadoop lance de nombreuses tâches (map, reduce, ...).
Certaines peuvent « planter » ou « ralentir ».

Le TaskTracker (MR v1) ou l'ApplicationManager (MR v2) monitore
fortement les exécutions des tâches, et détectent ces
« ralentissements ».



Hadoop peut faire une exécution spéculative : il lance de nouvelles instances des tâches « en retard », et dès qu'il obtient des résultats d'une tâche, il tue son doublon.

Mais cette démarche a un coût :

- en charge de calcul (création fréquentes de tâches redondantes)
- en déplacement de données (surtout si on redonne un *reducer*)

On peut débrayer ce comportement (ex : cluster HPC très fiable)

Technologies d'Hadoop

