TELECOM ParisTech

INSTITUT Mines-Télécom

2018/2019

INTRODUCTION

QUELQUES MOTS SUR VOTRE ENSEIGNANT

- Fabien PICHON
- Data Engineer à la Société Générale

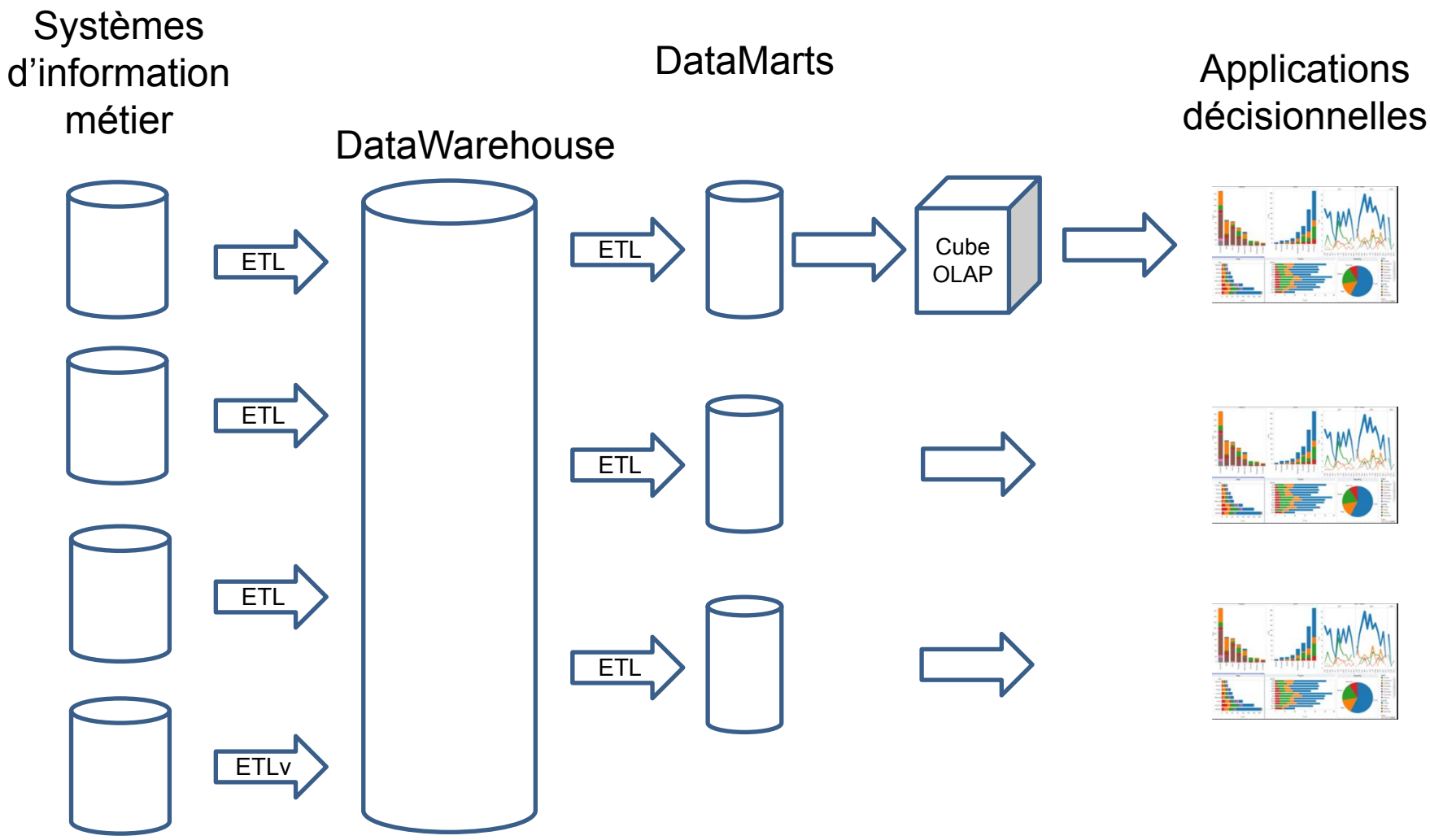
- fabien@grooptown.com

ORIGINES DES DATALAKES

Informatique décisionnelle et Data Warehousing

- Des applications silotées dans les grandes entreprises (facturation, achats, …)
- Un besoin de vision globale des données pour prendre des bonnes décisions
- Une réconciliation de ces données dans un Data Warehouse
- Des Datamarts pour exposer ces données à des applications décisionnelles
- Des cubes OLAP pour permettre de représenter des données relationnelles en n dimensions.
- Des outils de requête et de visualisation pour représenter des indicateurs selon plusieurs axes d'analyse.

EXEMPLE DATAWAREHOUSING



MODÉLISATION

- Le DataWarehouse et les DataMarts reposent sur des SGBD, des bases de données relationnelles
- Il est donc nécessaire de créer un modèle de donnée **au préalable.**
- Cela implique de représenter l'entreprise sous forme d'objets métiers qui n' évoluent que très peu.

- On a besoin de connaître notre **besoin** avant d'ingérer les données.

L'APPROCHE LAC DE DONNÉES

- Récupération de toute les données porteuses de sens **dans leur format original.**
  - Données tabulaires en provenance de SGBD
  - Données semi structurées (ex: json, xml)
  - Données textuelles
- Archivage actif, les données sont stockées et peuvent être exploitées telles quelles.

En comparaison:

- DataWarehouse: Intégration compliquée, requêtes facile
- DataLake : Intégration facile, requêtes compliquées

CE QU'ON VEUT FAIRE

Besoins d'une plateforme big data:

- Stocker de très gros volumes de données **sans contrainte de format.**
- Pouvoir augmenter facilement la taille de la plateforme, ne pas être limité par l'espace disponible.
- Avoir une tolérance à la panne, ne jamais avoir de perte de données.
- Pouvoir récupérer facilement des données générées rapidement.
- Avoir accès à un ensemble d'outils permettant de manipuler de très gros volumes de données.

TELECOM ParisTech

INSTITUT Mines-Télécom

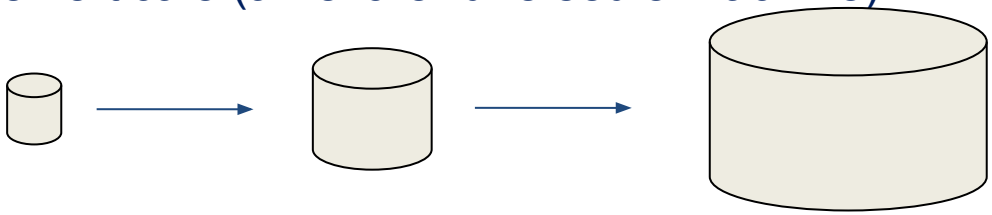
HADOOP ET SON ECOSYSTÈME

- Ensemble de technologies distribuées permettant de stocker et traiter de gros volumes de données (> To).
- Open Source
- Basé sur les publication de MapReduce, GoogleFS et BigTable de Google
- Développé initialement par Doug Cutting (Yahoo)
- Compte aujourd'hui ~100 commiters actifs (principalement chez Hortonworks)
- Scalabilité horizontale forte
- Tolérance à la panne
- Construit à la base pour tourner sur des machines peu coûteuses (plus vrai dans les faits maintenant)

SCALABILITÉ

Désigne la capacité d'un produit à s'adapter à un changement d'ordre de grandeur de la demande.

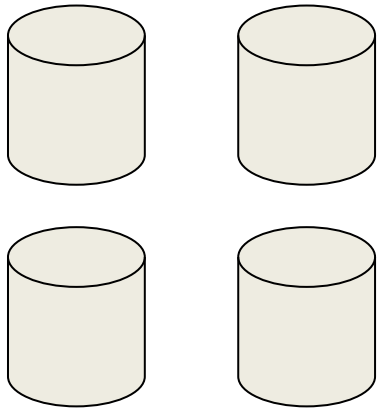
scalabilité verticale (améliorer une seule machine)



scalabilité horizontale (ajout de nouvelle machine)

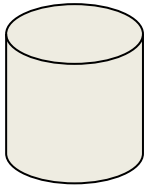
DÉFINITION : CLUSTER

Groupe de machines qui agit comme un seul et même système

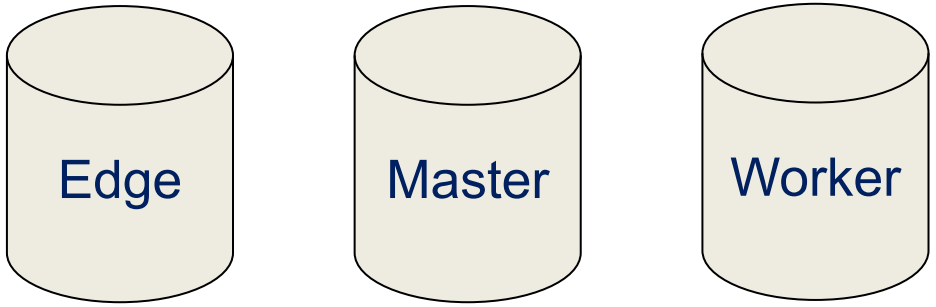


DÉFINITION : NOEUD

Un nœud est l'une des machines du cluster



Sur un cluster pur Hadoop, on distingue 3 types de noeuds

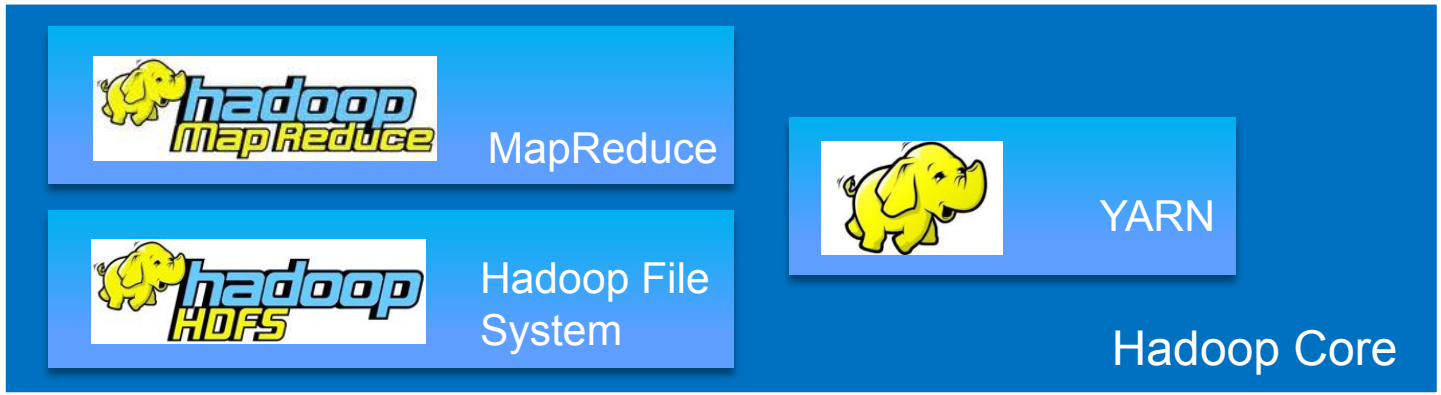


HADOOP CONCEPTS PRINCIPAUX

- **HDFS**: Un système de fichier distribué. Les données sont réparties et répliquées sur le cluster

- **MapReduce**: Un « paradigme de programmation ». C'est une façon de construire des applications distribuées. C'est aussi un framework intégré à Hadoop.

- **YARN:** Un gestionnaire de ressources. Permet de lancer des applications sur le cluster.



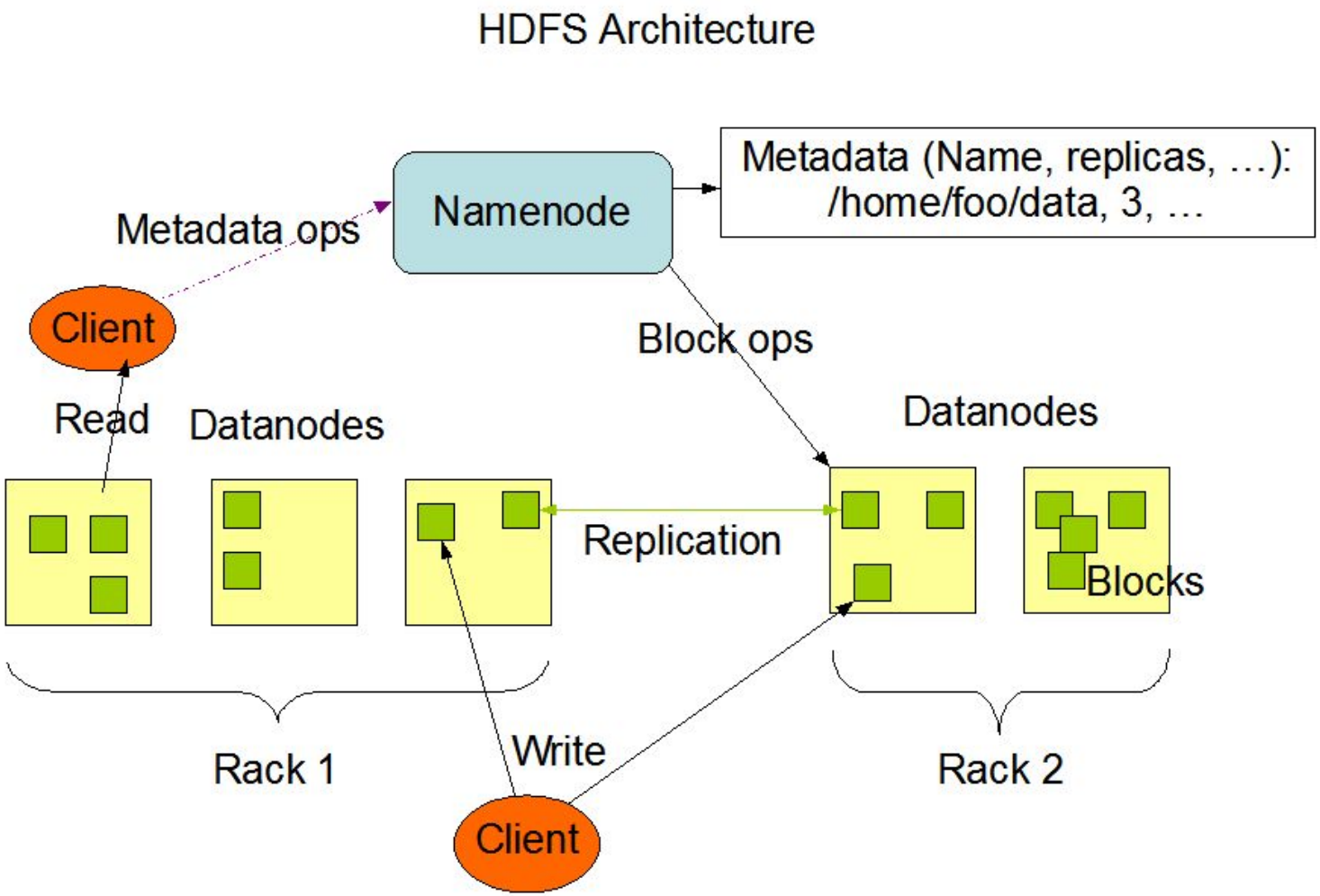
HDFS : PRÉSENTATION

- Hadoop Distributed File System : système de gestion de fichiers distribués dans Hadoop

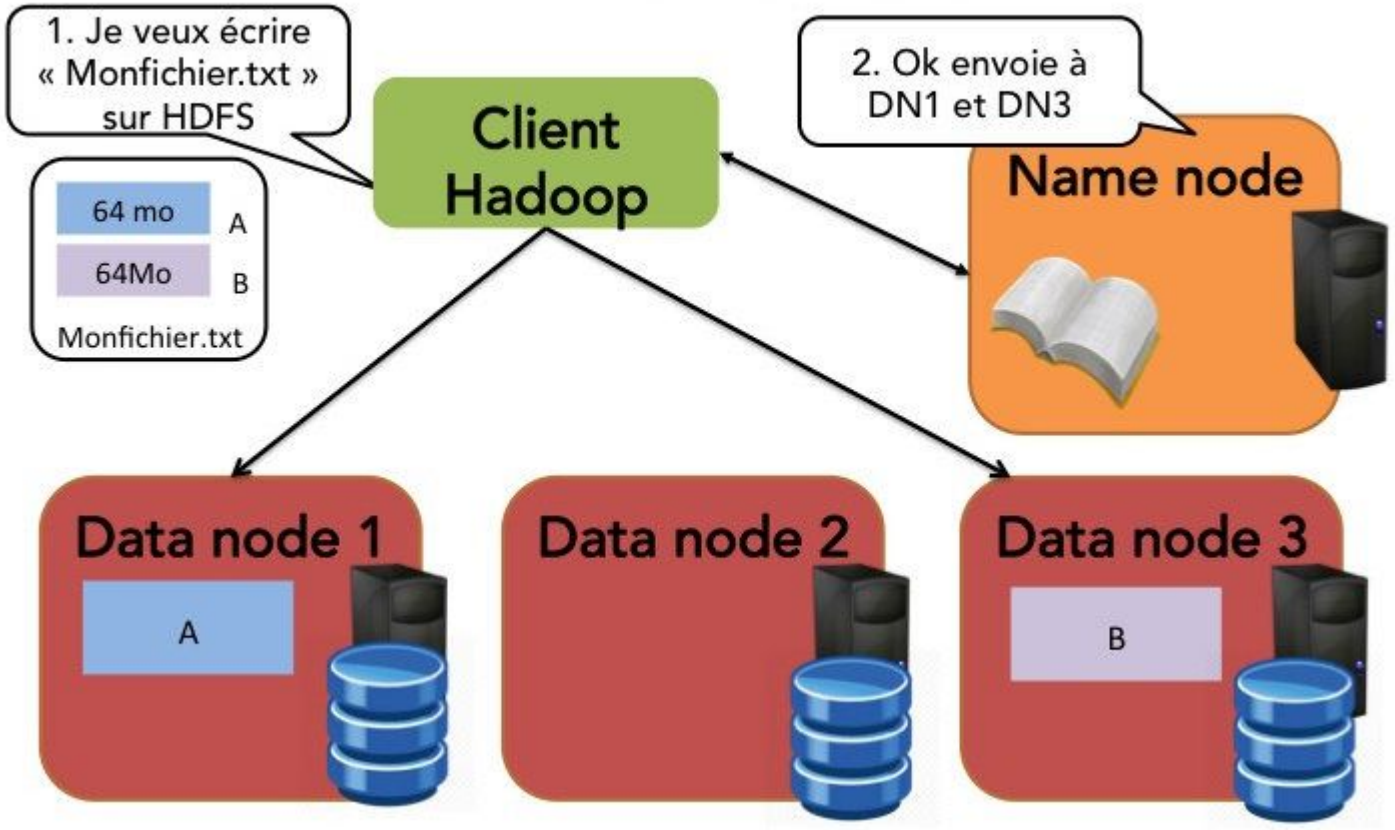
- principe de base : haute tolérances aux pannes (à la base fait pour fonctionner sur des machines peu coûteuses) - les données sont répliquées
- optimisé pour stocker des fichiers volumineux
- fonctionne sur Unix, développé en Java

- 2 composants :
  - Namenode : gardent un index des données
  - Datanode : écrivent/lisent les fichiers stockés

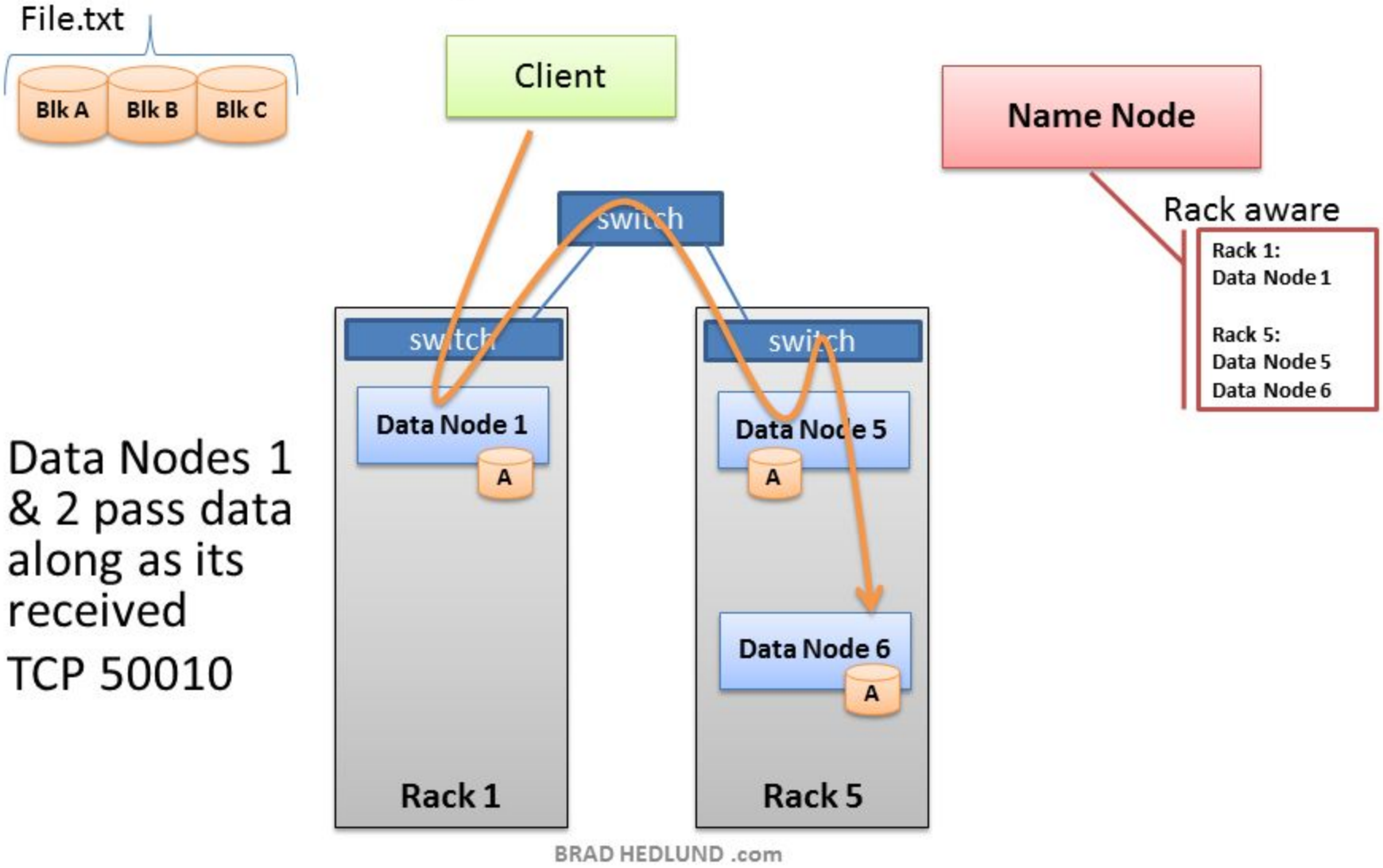
HDFS : FONCTIONNEMENT



Ecriture d'un fichier

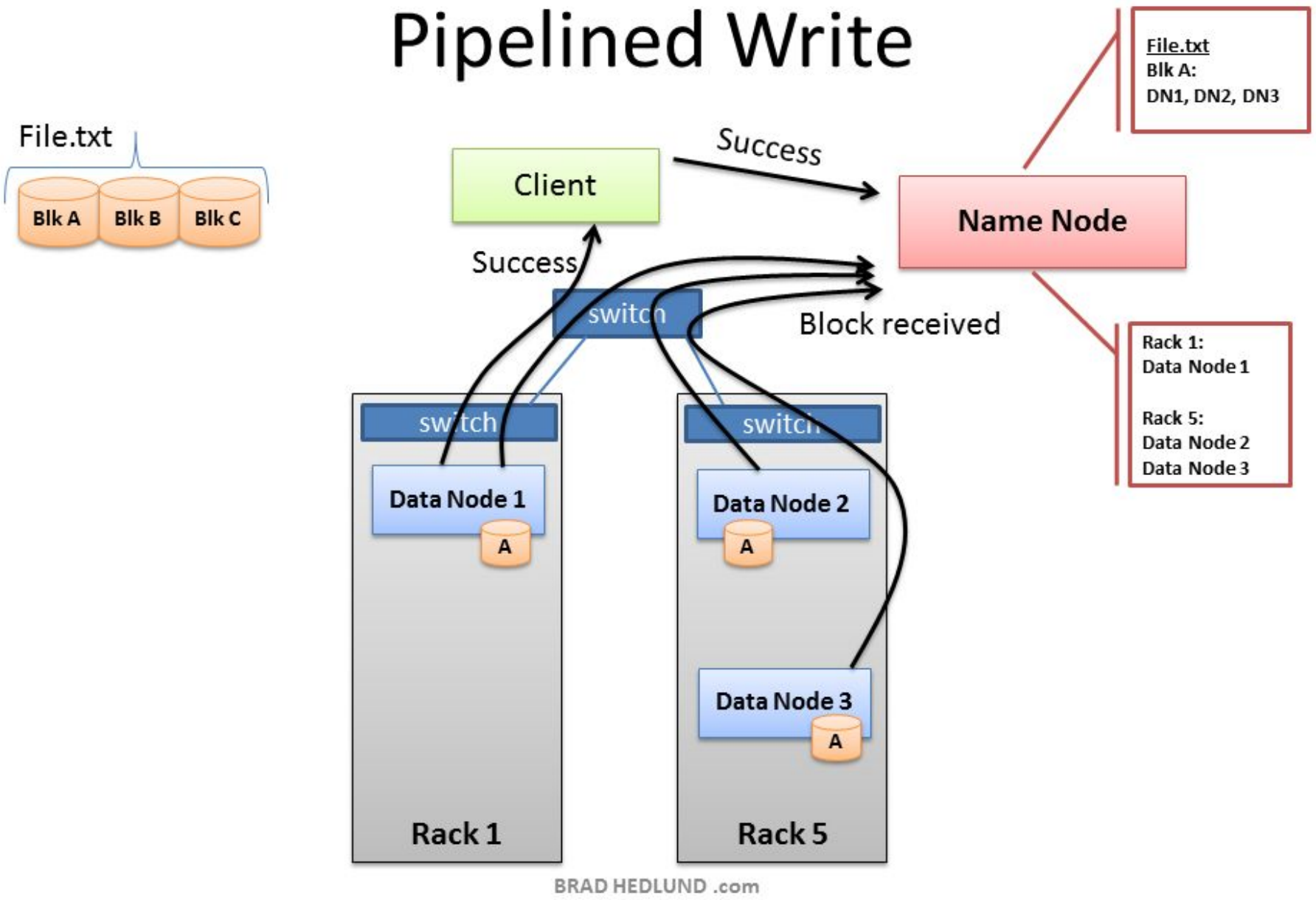


Pipelined Write

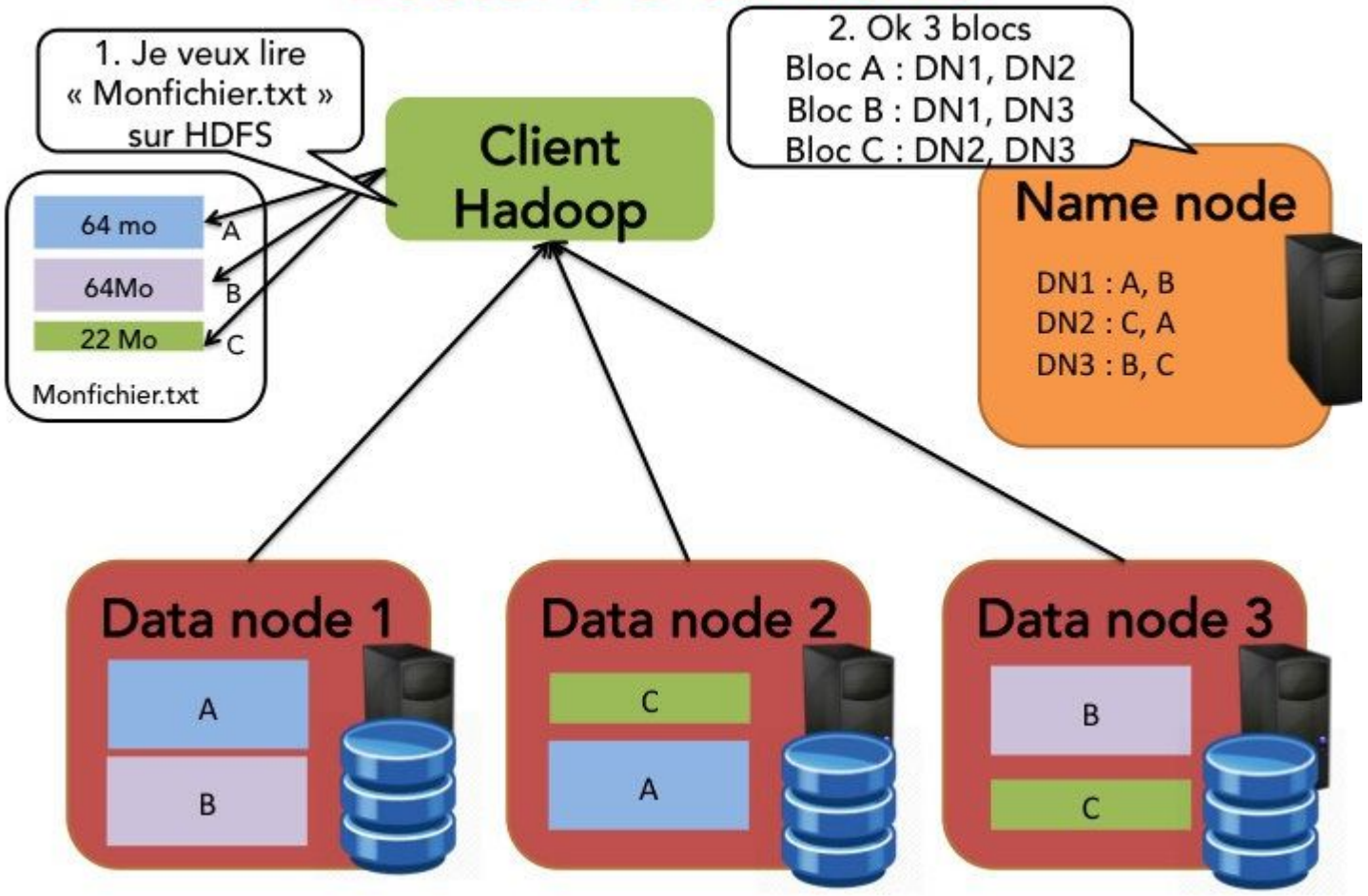


- Data Nodes 1 & 2 pass data along as its received
- TCP 50010

Pipelined Write



Lecture d'un fichier



HDFS : COMPLÉMENTS

- le facteur de réplication est de 3 par défaut
- la taille des blocks HDFS est de 64 Mo
  - stocker un fichier de 1 ko ou de 60 Mo demandera un bloc de 64 Mo dans HDFS. Passer les 64 Mo en demandera un nouveau
  - motivations : les requêtes dans HDFS sont cheres (rappatrient beaucoup de métadata). Séparer les données en blocs de 64 Mo est une optimisation sur les requêtes
  - cela amène à la question : comment stocker des petits fichiers dans Hadoop ? => Hadoop HAR, mais surtout éviter les petits fichiers !!
- Répartition intelligente (essaye de répartir la charge entre les machines)
- Stockage “rackaware” (stocke sur des racks différents)

Comprendre l'organisation / la gestion des metadata du namenode et le Checkpointing process

```
${dfs.namenode.name.dir}/
├── current
│   ├── VERSION
│   ├── edits_0000000000000000001-0000000000000000019
│   ├── edits_inprogress_0000000000000000020
│   ├── fsimage_0000000000000000000
│   ├── fsimage_0000000000000000000.md5
│   ├── fsimage_0000000000000000019
│   ├── fsimage_0000000000000000019.md5
│   └── seen_txid
└── in_use.lock
```

https://blog.csdn.net/weixin_40978095

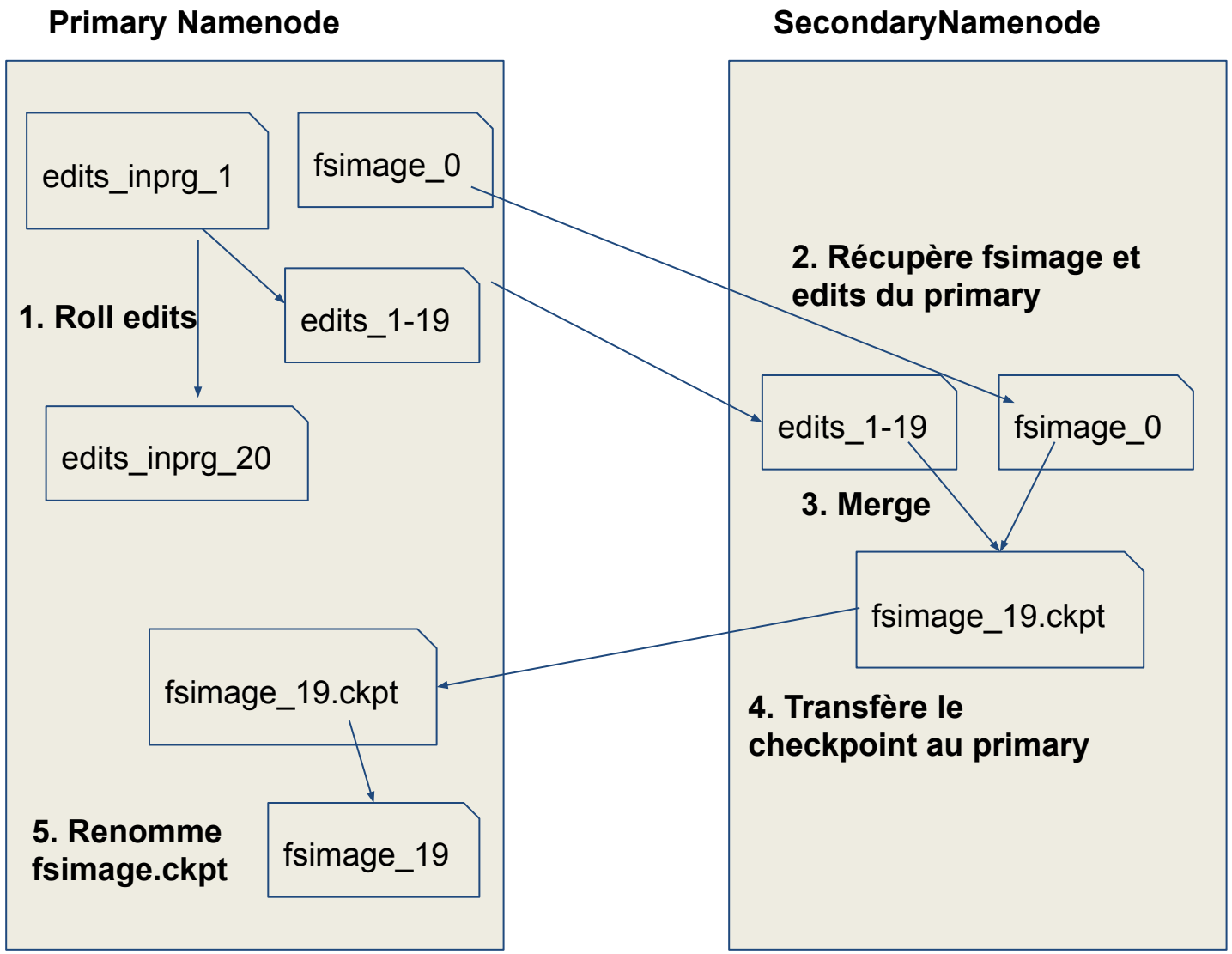
Comprendre l'organisation / la gestion des metadata du namenode et le Checkpointing process

FSImage:

- Niveau de réplication des fichiers
- Modification et access times
- Access permission
- Block size
- L'identifiant des blocks

L'association blocks / Datanodes est gardée en mémoire, et est construite en demandant aux datanodes leur liste de blocks lorsqu'ils rejoignent le cluster (et périodiquement ensuite pour conserver un état à jour du système)

Comprendre l'organisation / la gestion des metadata du namenode et le Checkpointing process

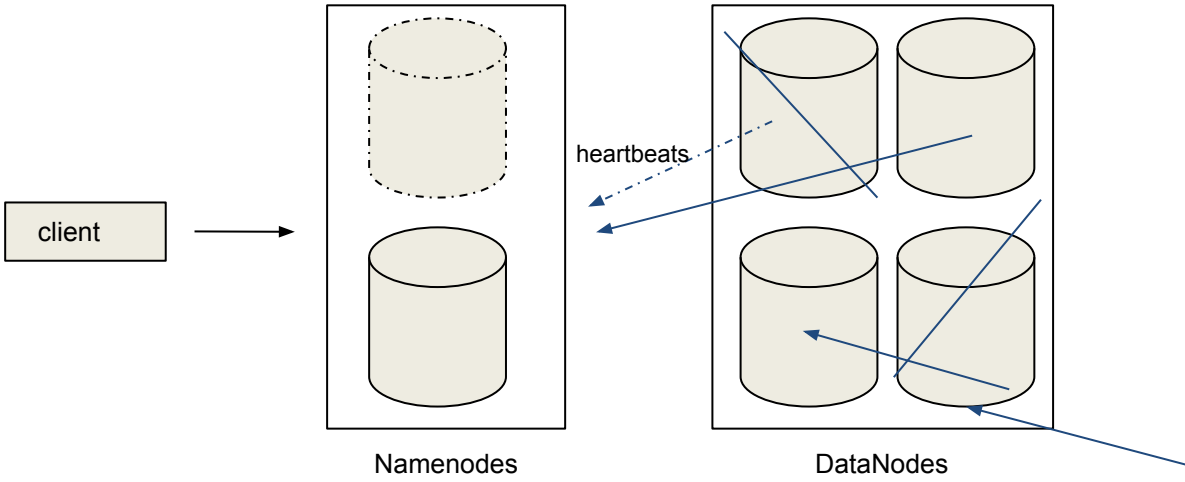


HDFS : CASE OF FAILURE DATANODE

Un datanode tombe

Il n'envoie plus de heartbeat aux namenodes

Si le namenode ne reçoit rien pendant une certaine durée, les blocs de données sont répliqués vers d'autres machines

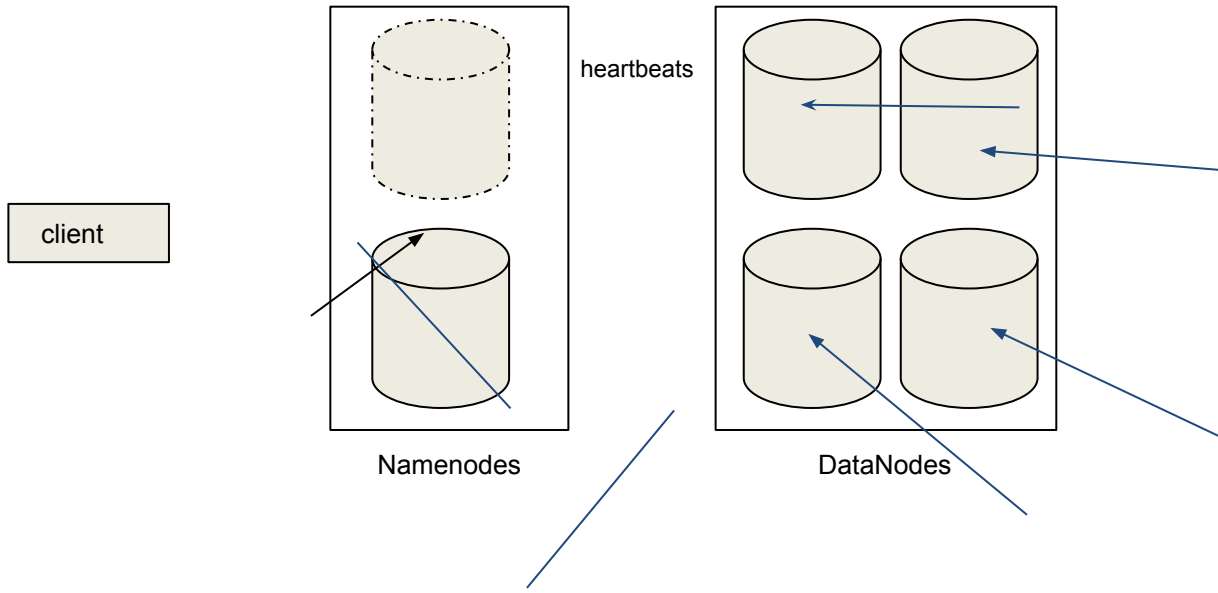


HDFS : CASE OF FAILURE NAMENODE

Un namenode tombe

Le client ne parvient plus à joindre le namenode principal, les demandes sont envoyées vers le namenode secondaire

Le namenode secondaire gère toutes les interactions avec le cluster à la place du primaire



HDFS : USAGE

HDFS n'est pas un système de fichier qu'on peut monter pour être accessible via les outils classiques de notre OS. (EXT4, NTFS, NFS, etc…)

On doit utiliser une interface pour intéragir avec HDFS.

Exemples:

- CLI: *hadoop fs ou hdfs dfs*
- Ambari File Viewer
- Hue
- WebHDFS, un web service REST

HADOOP FS

Exemple - put et get :

- Ecriture / récupération d'un fichier :
  - écriture du fichier foo.log dans /user/username/foo.log

```
$ hadoop fs -put foo.log foo.log
```

  - Récupération du fichier foo.log sur le post client

```
$ hadoop fs -get foo.log
```

HADOOP FS

Exemple - les outils GNU « classiques »:

- Les commandes GNU classiques :
  - mkdir : crée un répertoire sur hdfs

```
$ hadoop fs –mkdir /dir
```

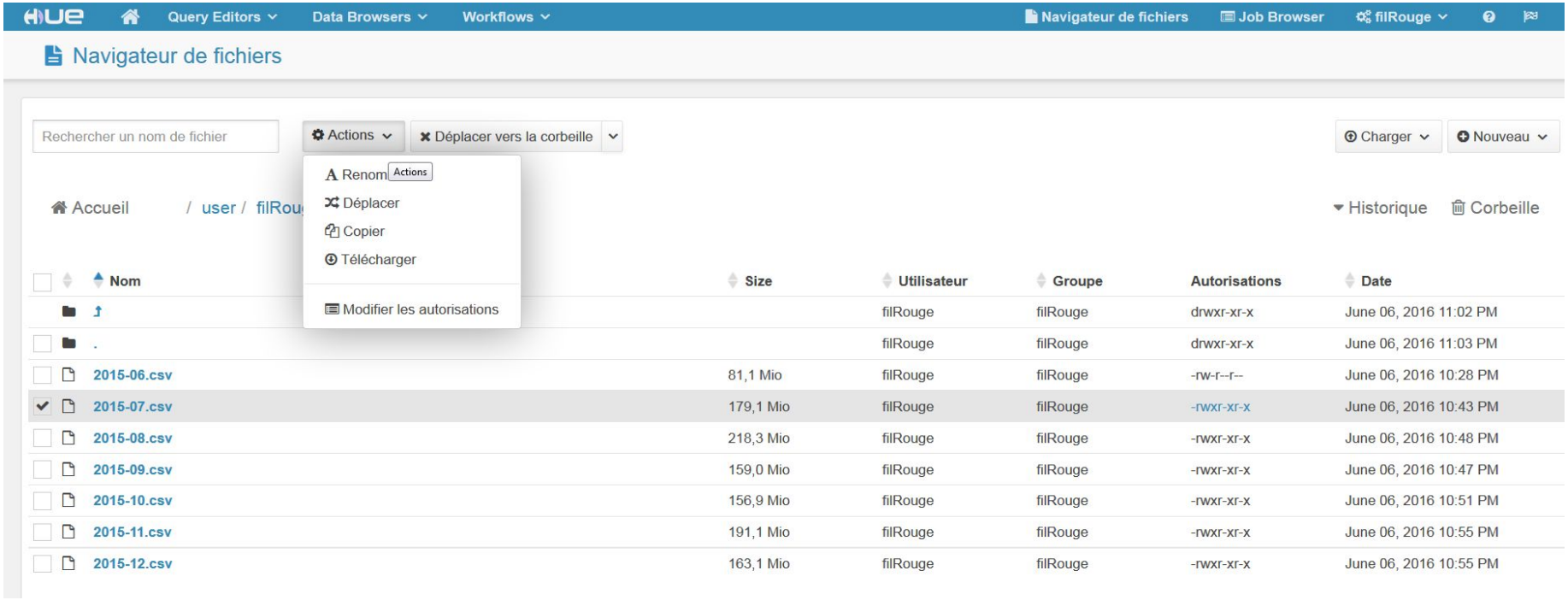
  - ls : affiche la liste des fichiers dans un répertoire hdfs

```
$ hadoop fs -ls /dir
```

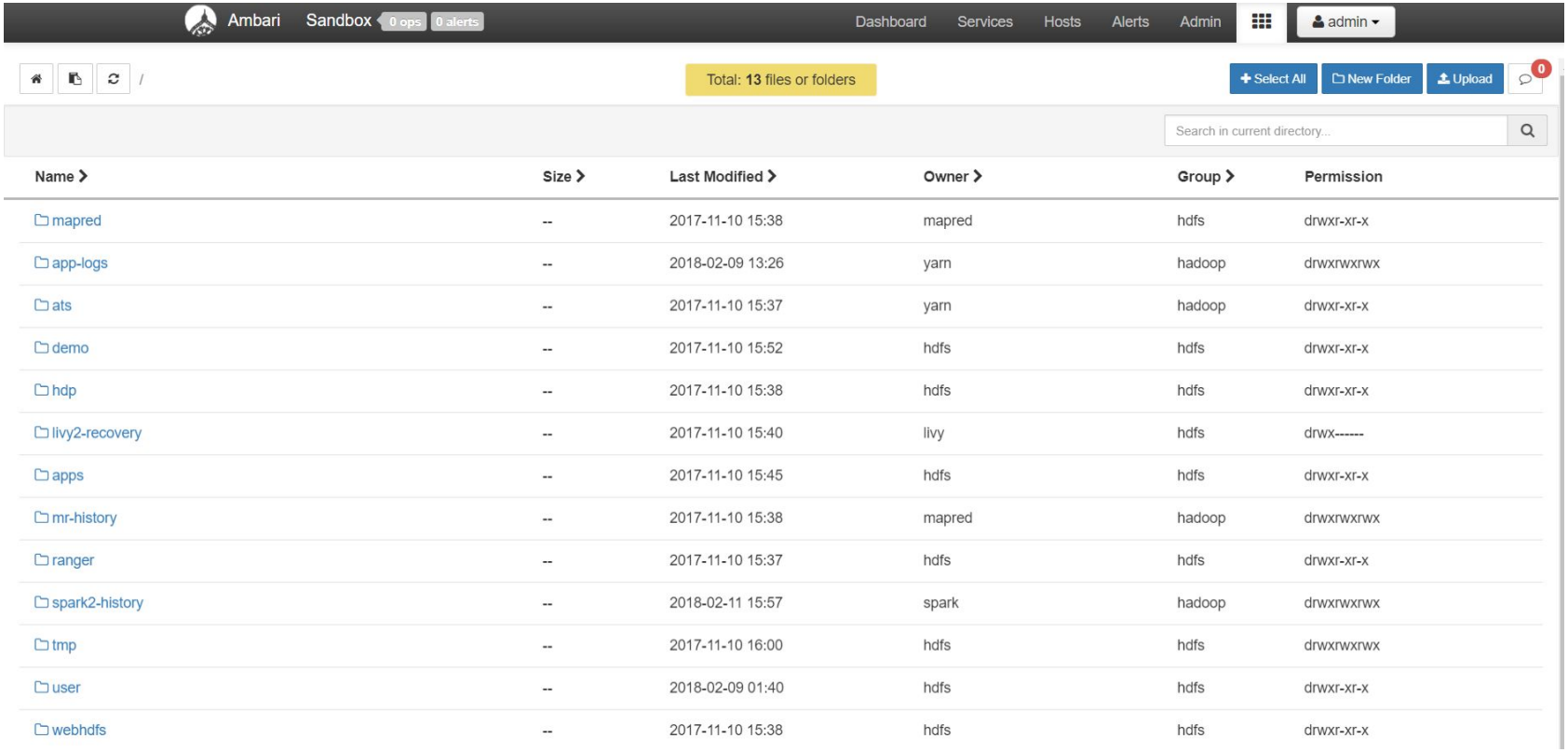
  - cat : affiche le contenu d'un fichier (attention à sa taille)

```
$ hadoop fs -cat /dir/foo.log
```

HUE


Query Editors
Data Browsers
Workflows
Navigateur de fichiers
Job Browser
filRouge
Navigateur de fichiers
Rechercher un nom de fichier
Actions
Déplacer vers la corbeille
Charger
Nouveau
Renom
Actions
Déplacer
Copier
Télécharger
Modifier les autorisations
Accueil
/ user / filRou
Historique
Corbeille
Nom
Size
Utilisateur
Groupe
Autorisations
Date
filRouge filRouge drwxr-xr-x June 06, 2016 11:02 PM
. filRouge filRouge drwxr-xr-x June 06, 2016 11:03 PM
2015-06.csv 81,1 Mio filRouge filRouge -rw-r--r-- June 06, 2016 10:28 PM
2015-07.csv 179,1 Mio filRouge filRouge -rwxr-xr-x June 06, 2016 10:43 PM
2015-08.csv 218,3 Mio filRouge filRouge -rwxr-xr-x June 06, 2016 10:48 PM
2015-09.csv 159,0 Mio filRouge filRouge -rwxr-xr-x June 06, 2016 10:47 PM
2015-10.csv 156,9 Mio filRouge filRouge -rwxr-xr-x June 06, 2016 10:51 PM
2015-11.csv 191,1 Mio filRouge filRouge -rwxr-xr-x June 06, 2016 10:55 PM
2015-12.csv 163,1 Mio filRouge filRouge -rwxr-xr-x June 06, 2016 10:55 PM

AMBARI FILES VIEW



INTRODUCTION À MAPREDUCE

Les concepts de MapReduce (général) :

- Le concept existe avant Hadoop, c'est une façon de traiter de la données efficace pour des traitements distribués en architecture
- Chaque nœud traite les données qu'il possède dans la mesure du possible
- Traitement en deux phases :
  - Map
  - Reduce
- Hadoop contient une API MapReduce pour traiter les données sur HDFS

INTRODUCTION À MAPREDUCE

Les features de MapReduce (Hadoop) :

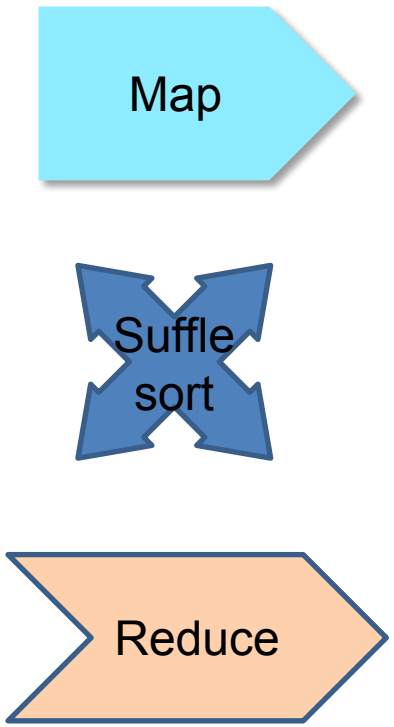
- Une parallélisation et distribution automatique des traitements
- Fault tolerance
- Une interface de programmation riche pour la création de programmes MapReduce
  - Les applications pures et dures MapReduce sont écrites en Java
  - Tout Hadoop core est écrit en Java (Hdfs, yarn, MapReduce)

- L'écriture de jobs mapreduce est fastidieuse. Des outils nous permettent de générer des jobs plus facilement.

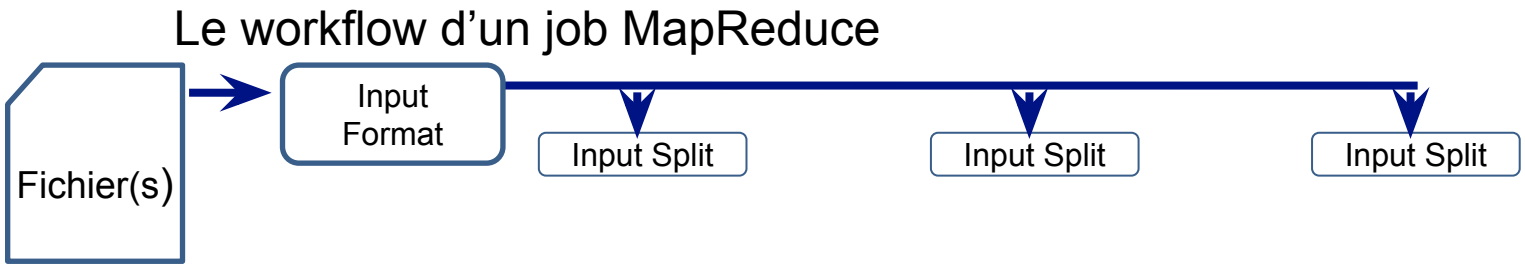
INTRODUCTION À MAPREDUCE

Les étapes clefs d'un job MapReduce

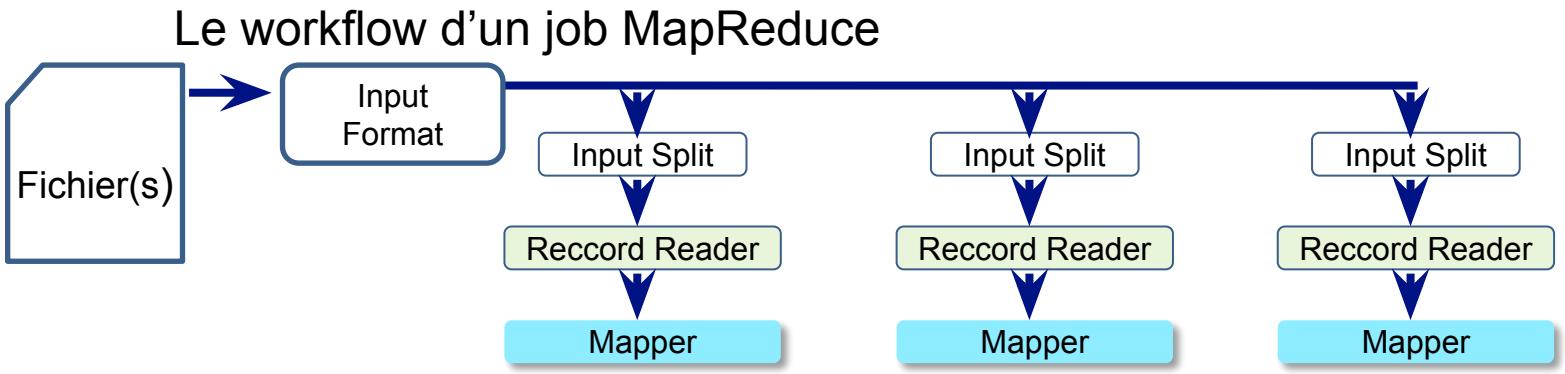
- La phase de Map (le mapper) :
  - Chaque tâche de Map traite un bloc d'HDFS
  - Dans la mesure du possible les tâches de map sont exécutées là où se trouve les données
- La phase de Shuffle and Sort :
  - Récupère les sorties des Maps
  - Trie et agrège les résultats pour la préparation de la phase Reduce
- La phase de Reduce (le reducer) :
  - Récupère les sorties du Shuffle & Sort
  - Génère les résultats finaux



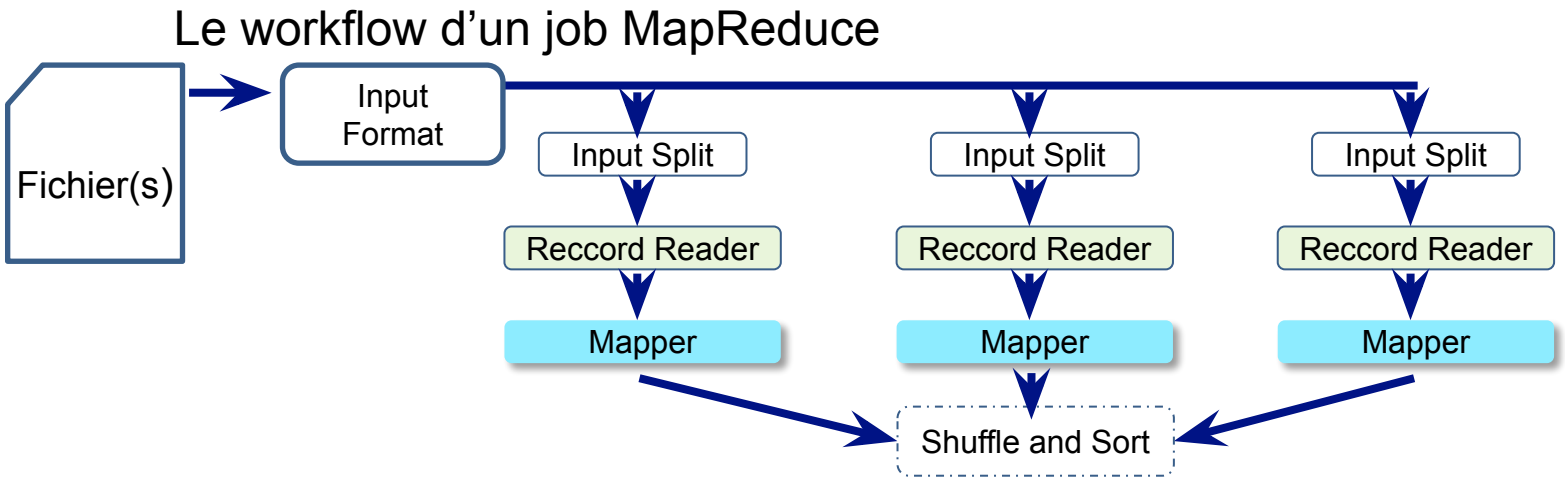
INTRODUCTION À MAPREDUCE



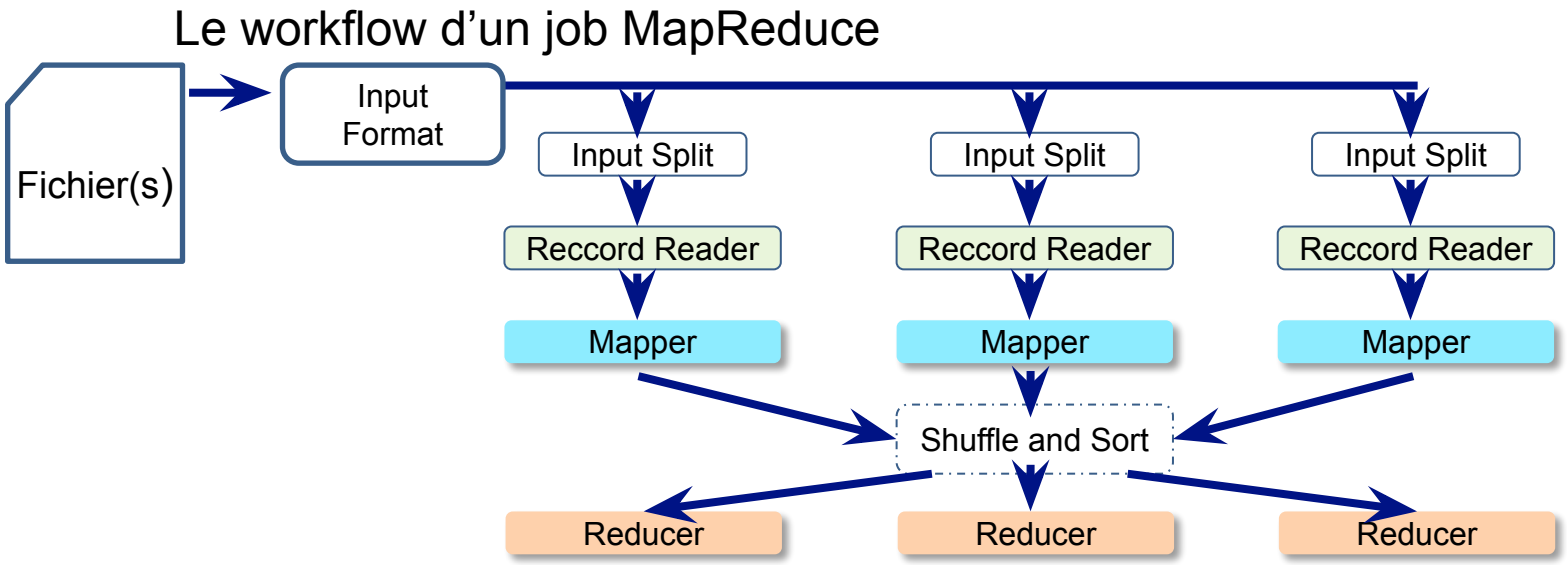
INTRODUCTION À MAPREDUCE



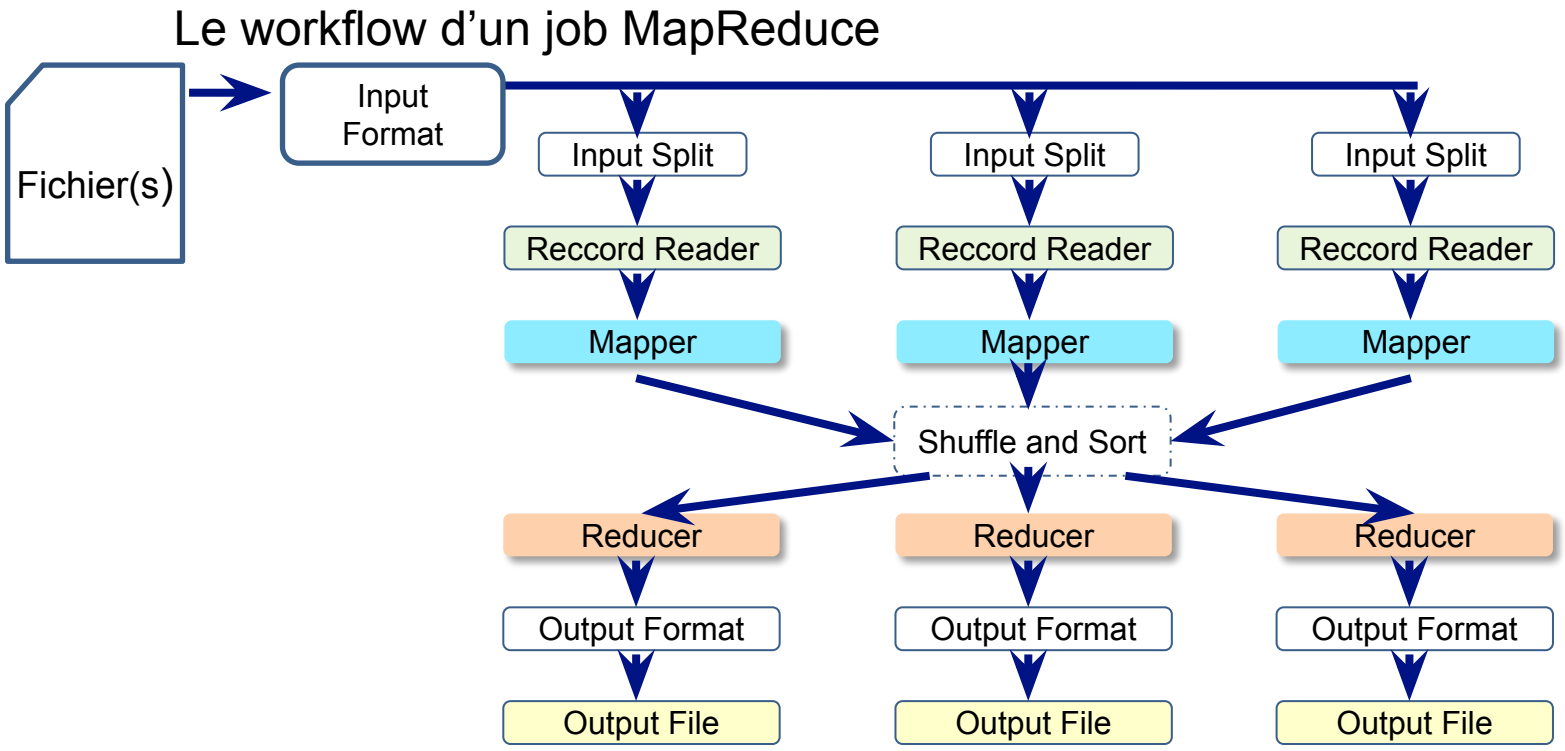
INTRODUCTION À MAPREDUCE



INTRODUCTION À MAPREDUCE



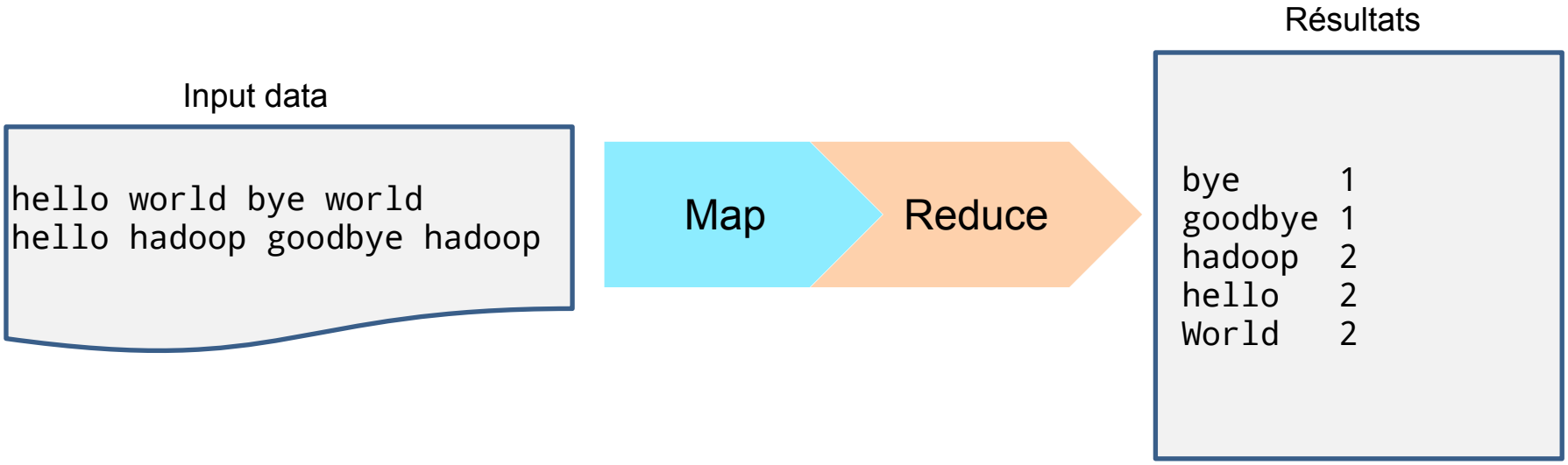
INTRODUCTION À MAPREDUCE



INTRODUCTION À MAPREDUCE

Le workflow d'un job MapReduce

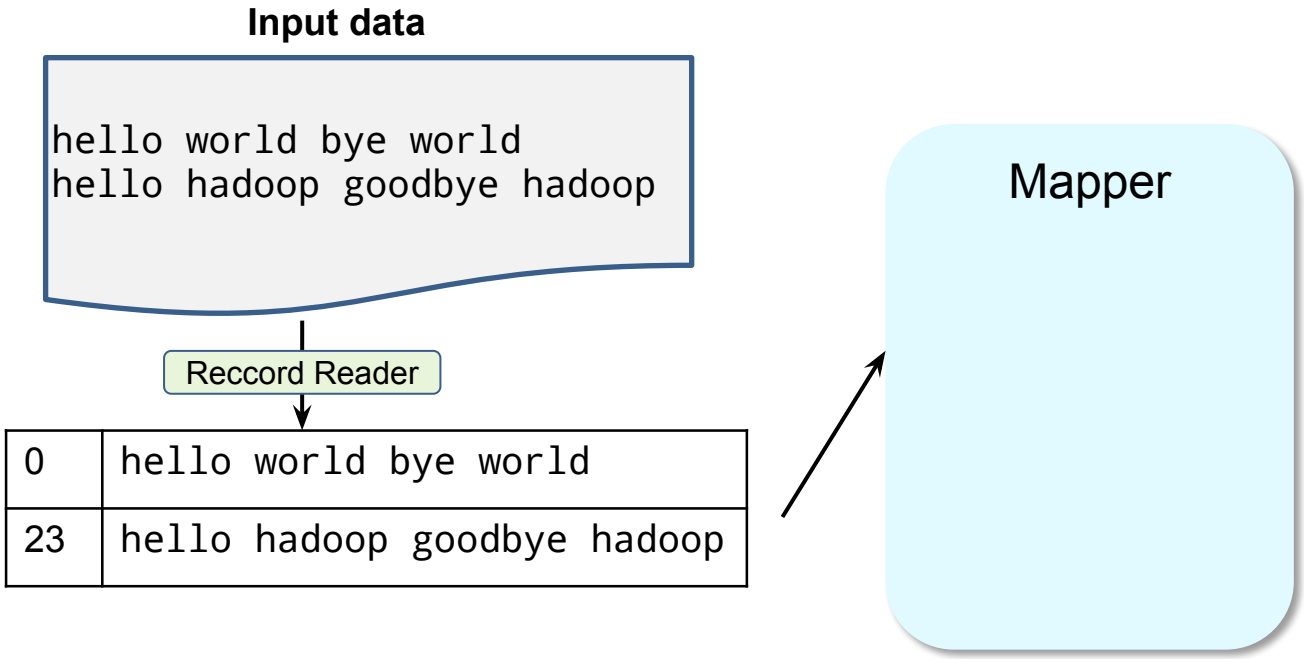
- Exemple : Wordcount (Hello World d'Hadoop)



INTRODUCTION À MAPREDUCE

Le workflow d'un job MapReduce - le Mapper

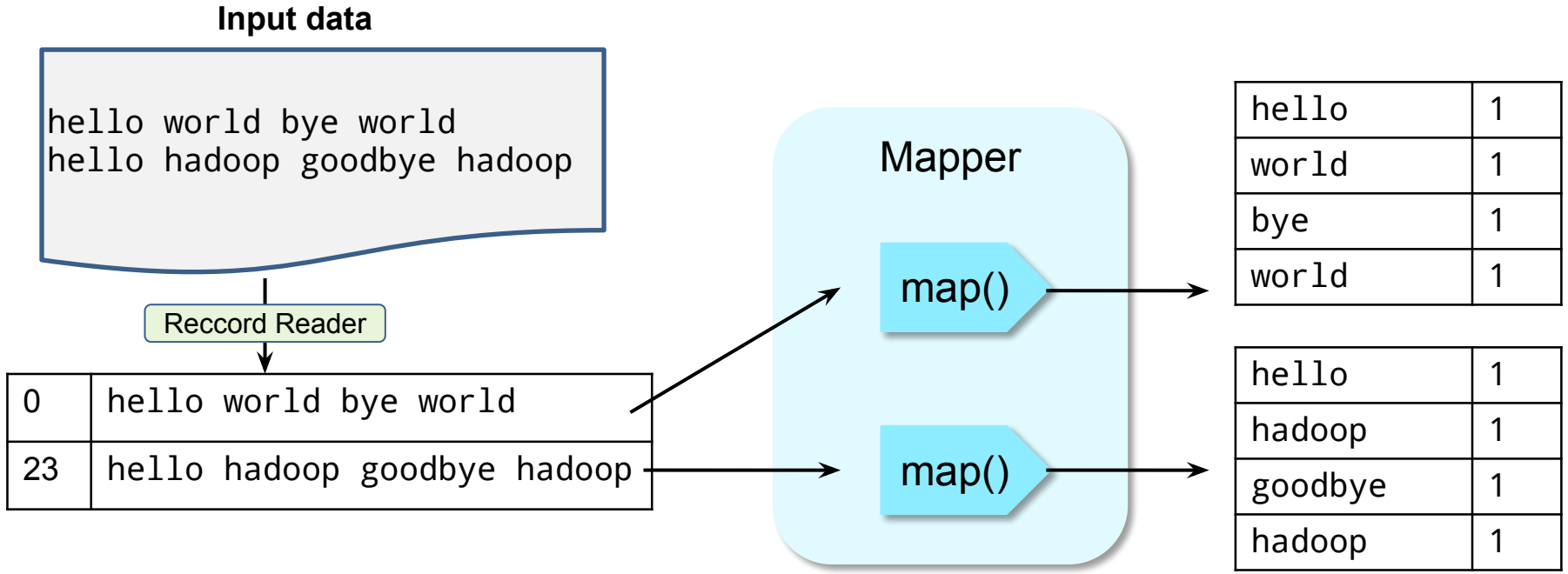
- Exemple : Wordcount (Hello World d'Hadoop)
- Le Mapper



INTRODUCTION À MAPREDUCE

Le workflow d'un job MapReduce - le Mapper

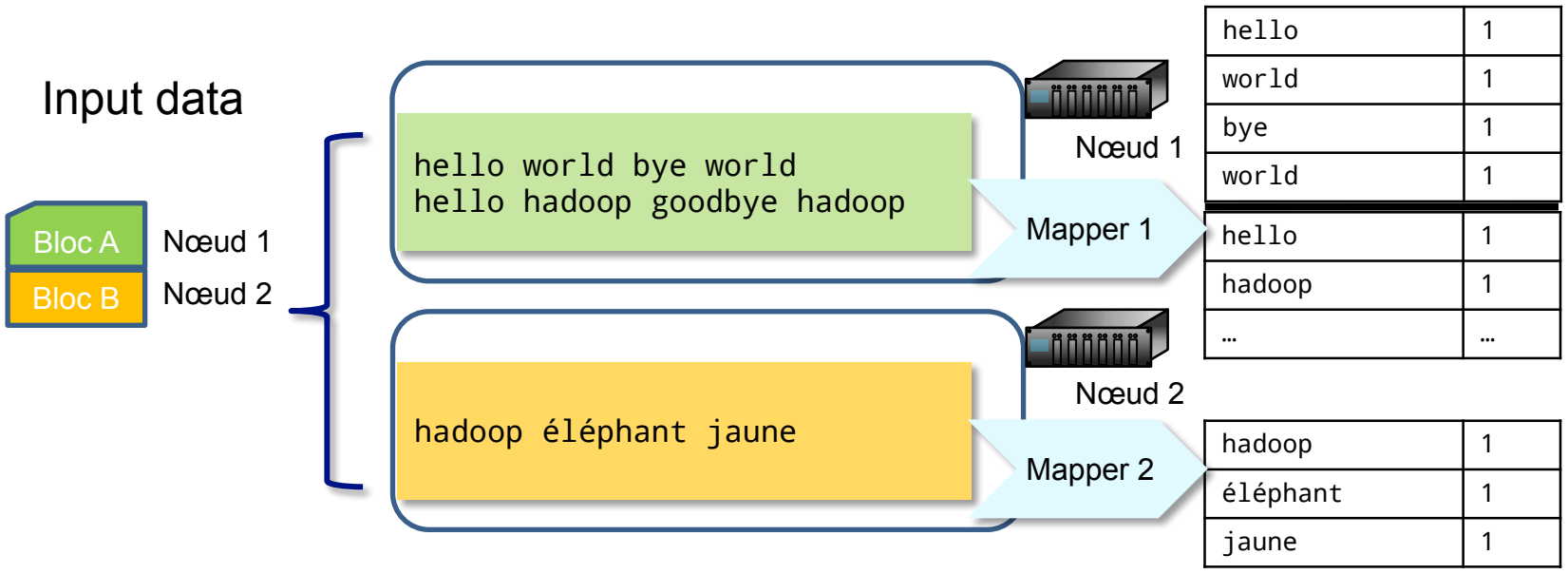
- Exemple : Wordcount (Hello World d'Hadoop)
- Le Mapper (lance autant de fois la fonction map() que de sorties du Reccord Reader)



INTRODUCTION À MAPREDUCE

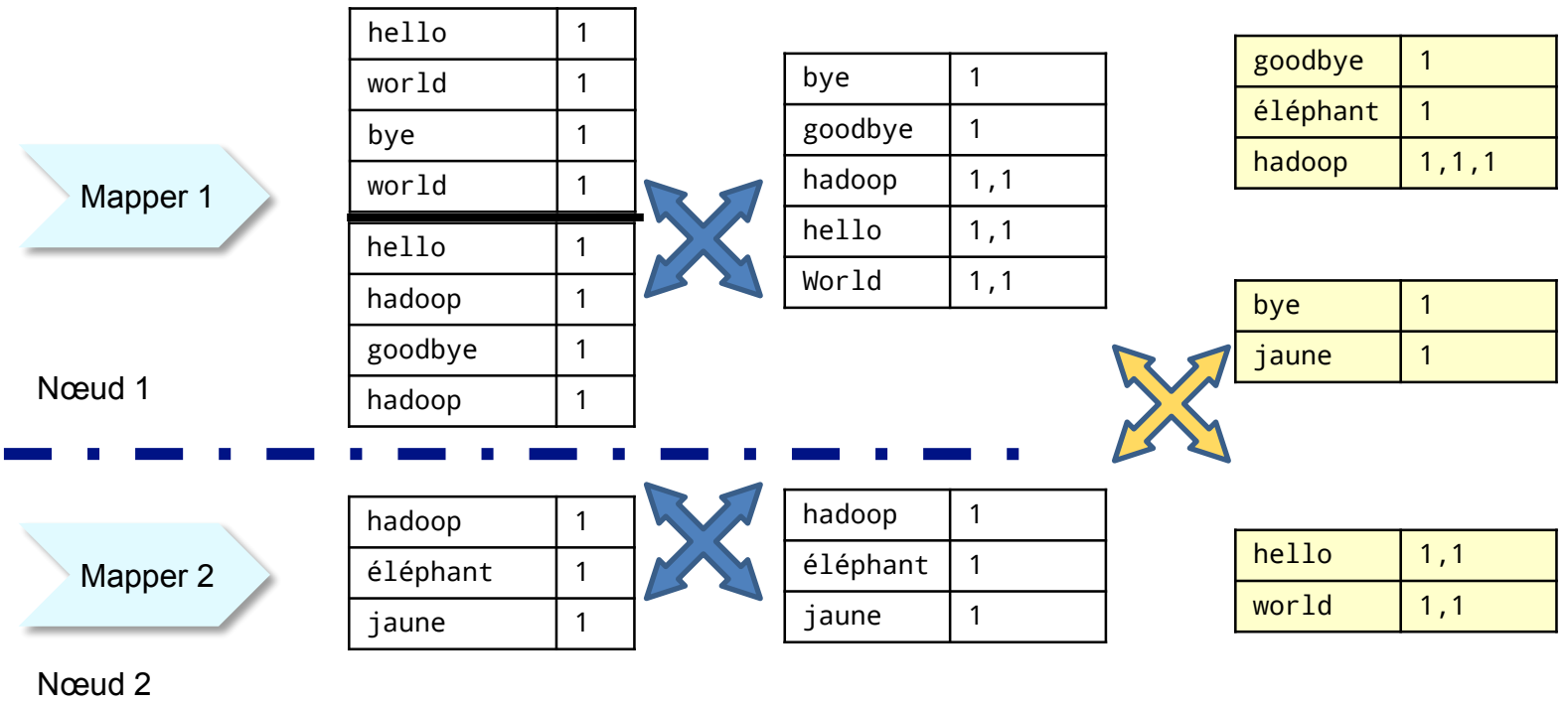
Le workflow d'un job MapReduce - le Mapper

- Le nombre de Mappers lancés en parallèle est égale au nombre de blocs en input
- Nombre de Mappers maximum = Totale du nombre de cœurs sur l'ensemble des nœuds



INTRODUCTION À MAPREDUCE

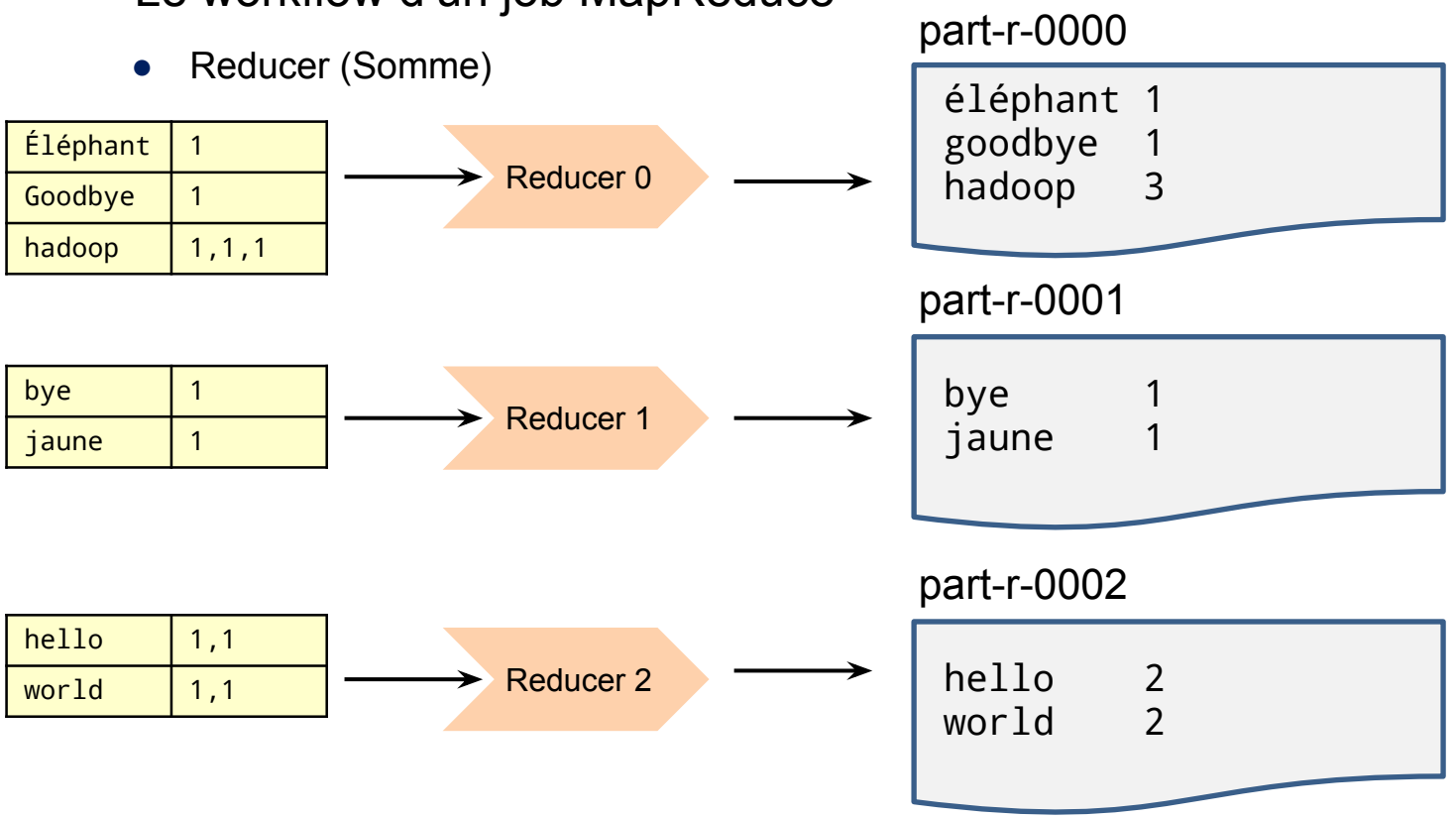
Le workflow d'un job MapReduce - Shuffle and Sort



INTRODUCTION À MAPREDUCE

Le workflow d'un job MapReduce

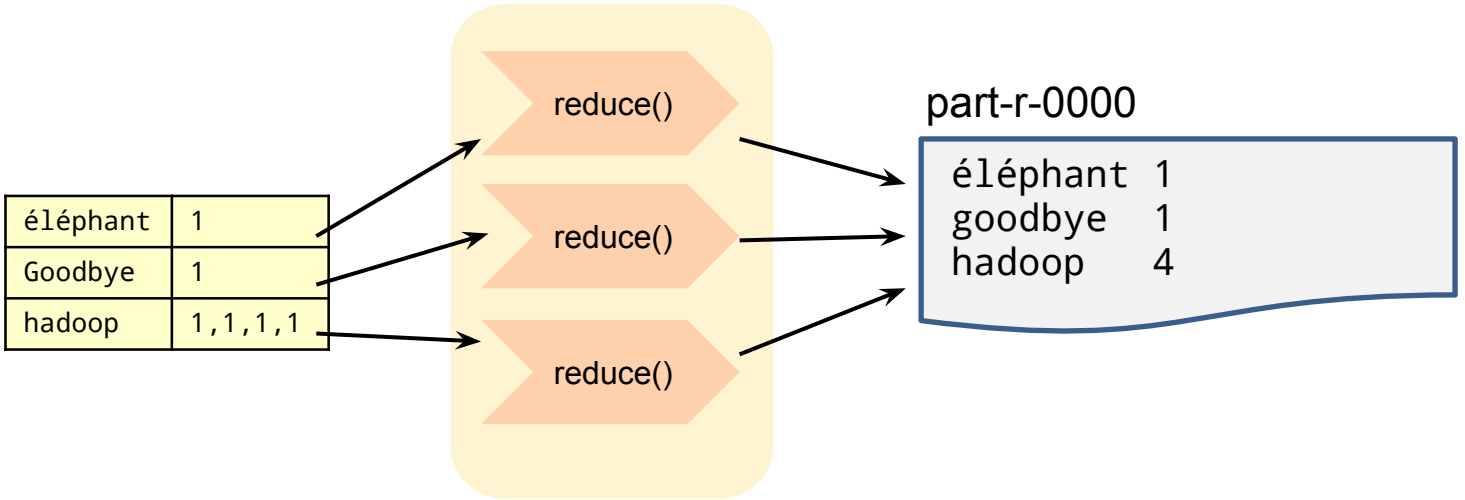
- Reducer (Somme)



INTRODUCTION À MAPREDUCE

Le workflow d'un job MapReduce

- Reducer (Somme)



INTRODUCTION À MAPREDUCE

l'intérêt de compter des mots (wordcount) :

- Problème complexe sur des données massives :
  - En mono-nœud, très consommateur en temps
  - Le nombre de mots uniques peut faire exploser la mémoire
- Le problème s'adapte bien sur le FrameWork MapReduce
  - De manière générale, les max, min, somme, count…
- On le retrouve dans les problématiques de Text-mining

INTRODUCTION À MAPREDUCE

Conclusion

Un programme MapReduce est composé de deux briques majeures à développer : un Mapper et un Reducer

- Un mapper prend l'input d'un fichier et génère des données intermédiaires sous la forme de clef/valeurs
  - Souvent parse, filtre, ou transforme les données du fichier
- Un reducer traite les outputs du mapper (après être passés par le Shuffle and Sort)
  - Souvent agrégation des données (Somme, Moyenne, Max, etc. …)

L'ÉCOSYSTÈME D'HADOOP

LE FONCTIONNEMENT D'UN JOB ET FOCUS SUR YARN

Le lancement d'un job sur Hadoop

- Une fois les classe Mappers et Reducer écrites
- Ecrire une classe « **Driver** » pour la configuration du job et les informations de lancement
- Compiler les classes :

```
$ javac -classpath `hadoop classpath` MyMapper.java
MyReducer.java Mydriver.java
```

- Créer le jar

```
$ jar -cvf MyMR.jar MyMapper.class MyReducer.class
MyDriver.class
```

- Lancer le job sur le cluster

```
$ hadoop jar MyMR.jar MyDrive input_file output_file
```

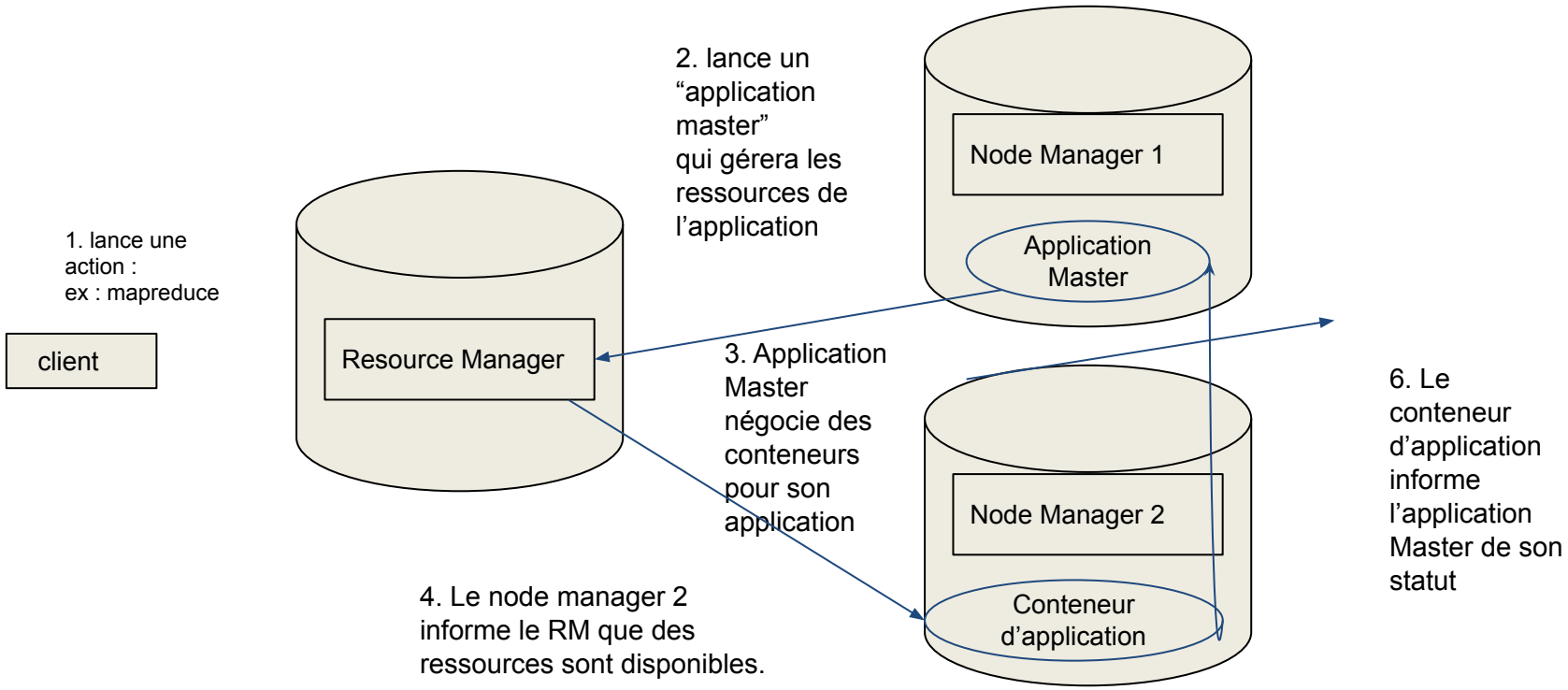
YARN : PRÉSENTATION

- Yet Another Resource Negotiator : système de gestion de ressources dans Hadoop
- Yarn gère l'allocation des ressources RAM, CPU, network pour chaque application lancée dans Hadoop
- 3 composants principaux :
  - Resource Manager (RM) : gère l'ensemble des ressources pour tout le cluster.
  - Node Manager : monitore les ressources par machine, reporte les informations au RM
  - Application Master (AM) : gère les ressources pour une application, négocie ses ressources au RM

YARN : CONTAINERS

- YARN lance des **containers**.
- Un containers est représenté par un espace sur le FS du nœud sur lequel il est exécuté.
- Avec Map Reduce par exemple, chaque mappers et reducers seronts exécutés dans des containers séparés.
- Dans le container on retrouve les script et les binaires nécessaires à l'exécution de la tâche.
- L'état du container est envoyé au ressource manager.

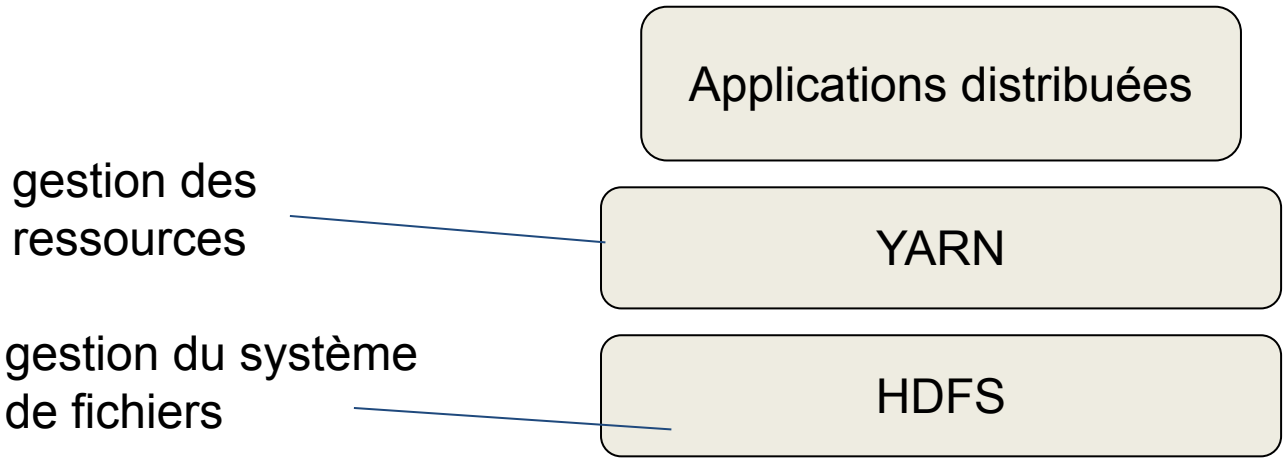
YARN : FONCTIONNEMENT



YARN : NOTES

- on peut définir des queues de priorité dans YARN
  - les applications exécutées avec une queue de haut niveau de priorité pourront détruire des conteneurs d'applications avec un faible niveau
  - on peut associer des % de ressources maximales à chaque queues
- YARN - long running application : les bases de données dans Hadoop sont aussi gérées par Yarn (exemple : HBase)
- YARN - label : on peut associer des labels aux node managers et associer des queues à une partie du cluster seulement
- Le futur de YARN : l'idée est que toute application, y compris un serveur web puisse tourner dans dans un docker embarqué dans un conteneur YARN

EN RESUMÉ


Applications distribuées
gestion des ressources
YARN
gestion du système de fichiers
HDFS

TP