

Big Data

GL4 (Option Management des Systèmes d'Information) - 2017

---

# Chp2 – Hadoop et MapReduce

Architecture et Comportement

---

# Hadoop

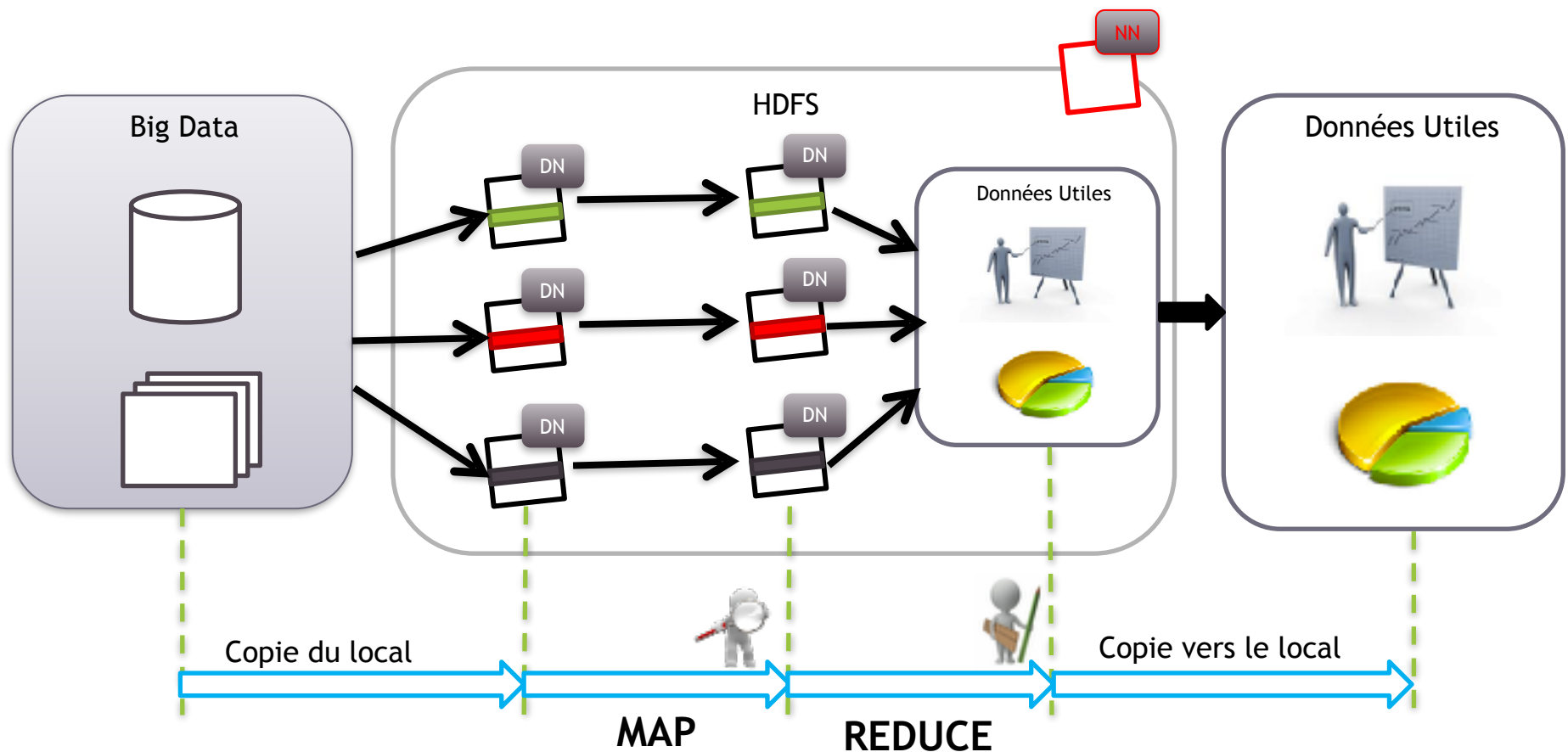
## Présentation du Framework

---

- 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, appelées 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

# Hadoop, HDFS et MapReduce

## Présentation du Framework

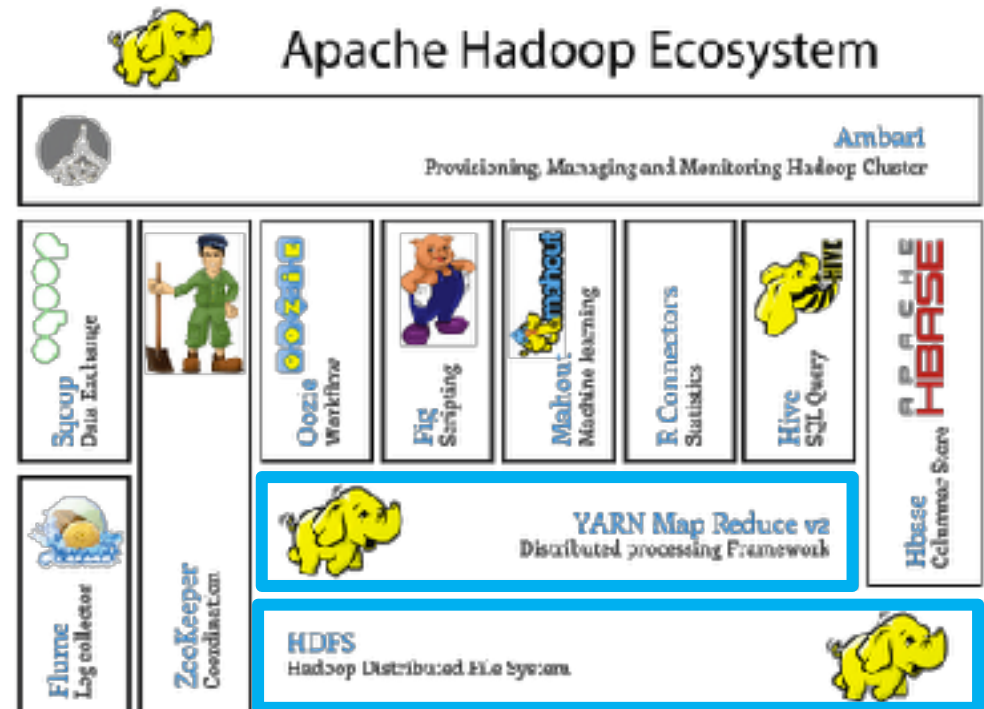


# Ecosystème de Hadoop

## Présentation du Framework

En plus des briques de base Yarn  
Map Reduce/HDFS, 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



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

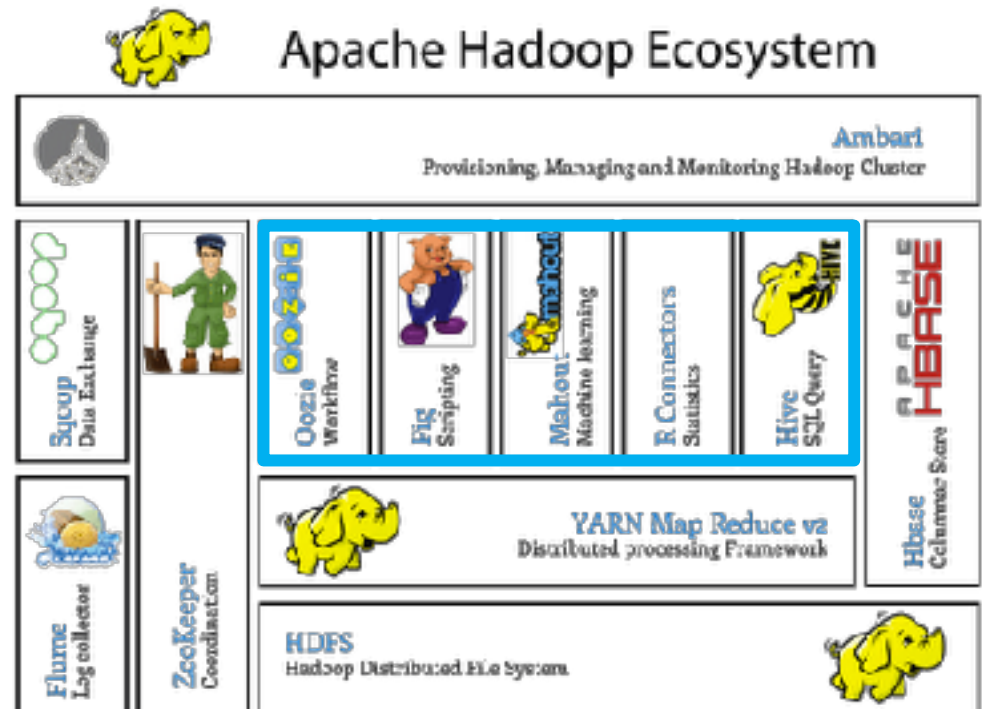
# Ecosystème de Hadoop

## Présentation du Framework

Parmi ces outils, certains se trouvent au dessus de la couche Yarn/MR, tels que:

- Pig: Langage de script
- Hive: Langage proche de SQL (Hive QL)
- R Connectors: permet l'accès à HDFS et l'exécution de requêtes Map/Reduce à partir du langage R
- Mahout: bibliothèque de machine learning et mathématiques
- Oozie: permet d'ordonnancer

les jobs Map Reduce, en définissant des workflows

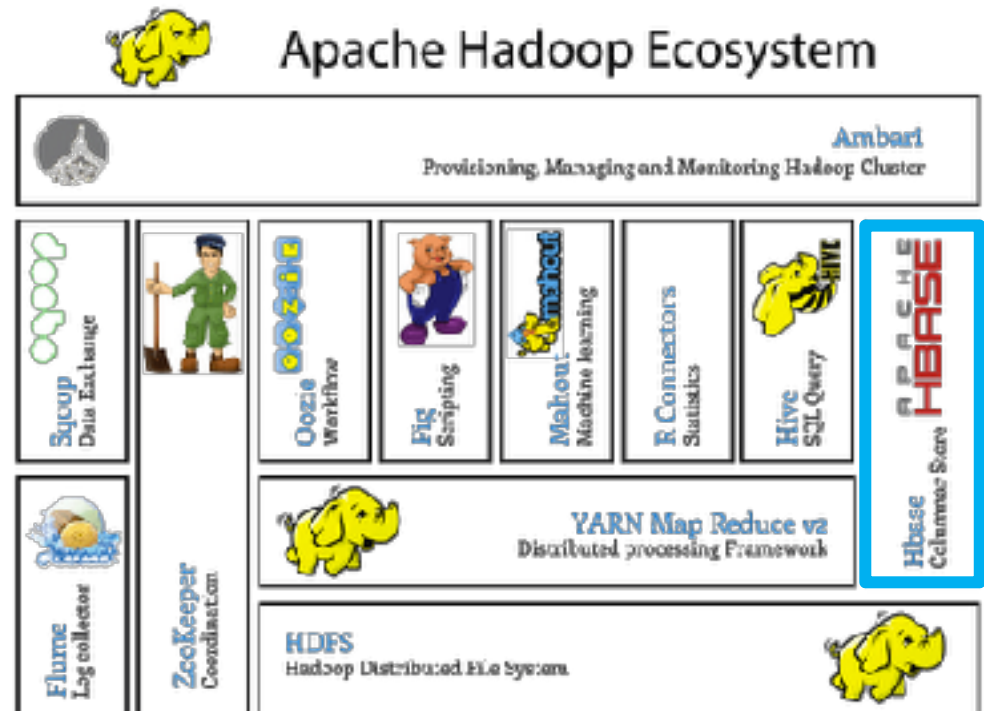


# Ecosystème de Hadoop

## Présentation du Framework

D'autres outils sont directement au dessus de HDFS, tels que :

- Hbase : Base de données NoSQL orientée colonnes
- Impala (pas représenté dans la figure): Permet le requêtage de données directement à partir de HDFS (ou de Hbase) en utilisant des requêtes Hive SQL

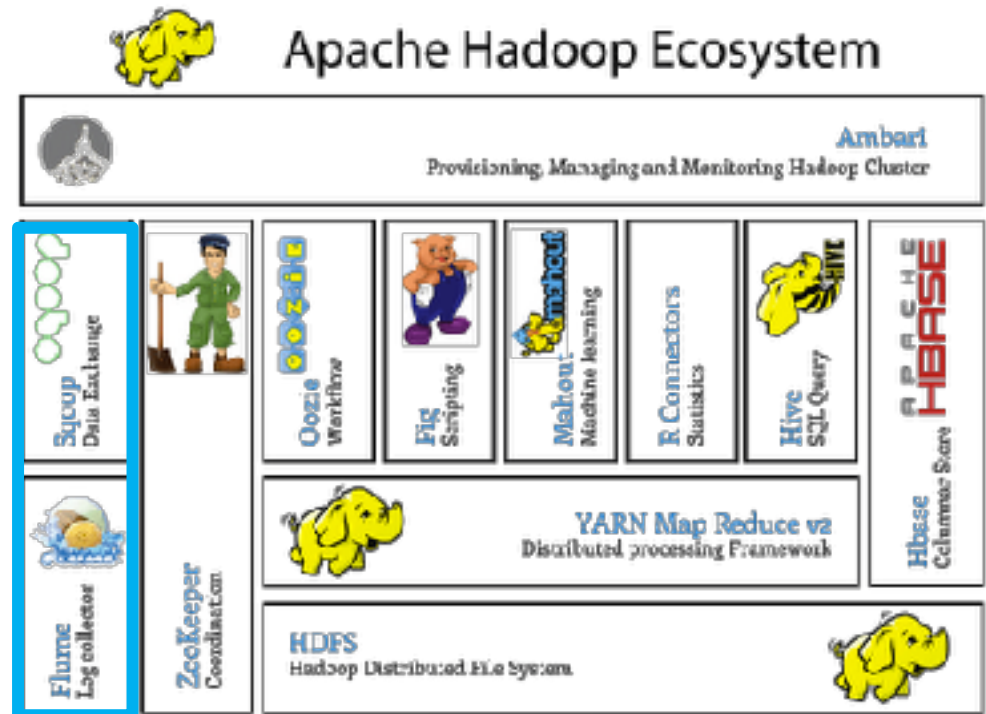


# Ecosystème de Hadoop

## Présentation du Framework

Certains outils permettent de connecter HDFS aux sources externes, tels que:

- Sqoop: Lecture et écriture des données à partir de bases de données externes
- Flume: Collecte de logs et stockage dans HDFS

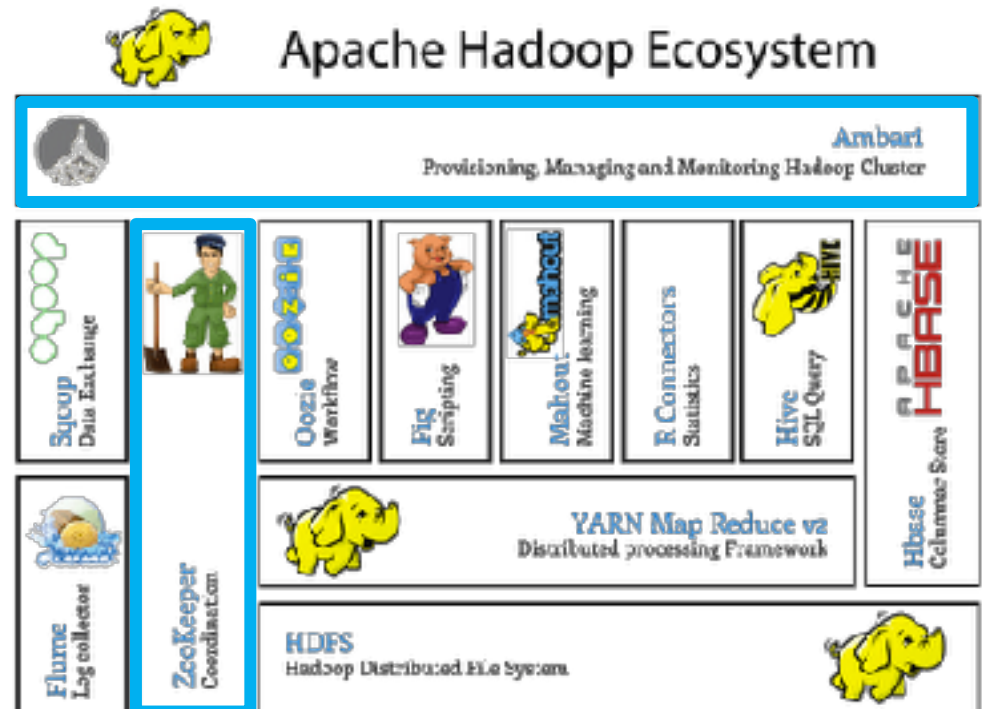


# Ecosystème de Hadoop

## Présentation du Framework

Enfin, d'autres outils permettent la gestion et administration de Hadoop, tels que:

- Ambari: outil pour le provisionnement, gestion et monitoring des clusters
- Zookeeper: fournit un service centralisé pour maintenir les information de configuration, de nommage et de synchronisation distribuée





---

Hadoop & Map Reduce

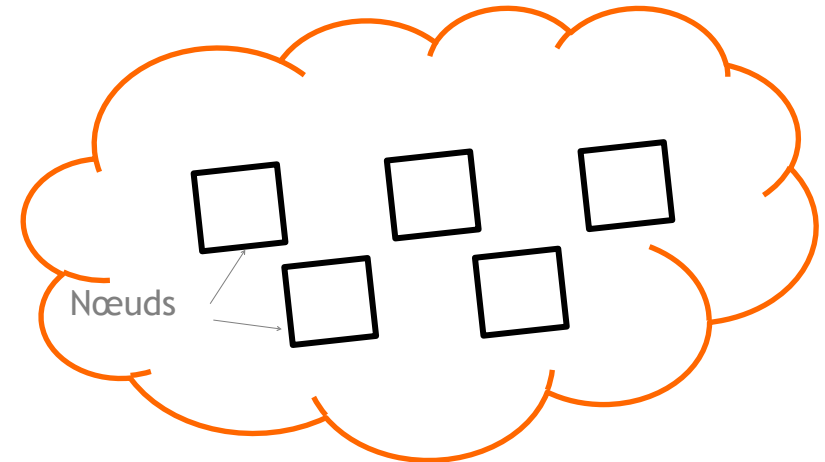
# HDFS: HADOOP DISTRIBUTED FILE SYSTEM

---

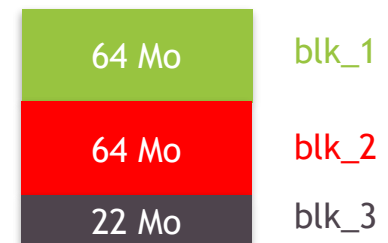
# HDFS : Hadoop Distributed File System

## Architecture

- HDFS est un système de fichiers distribué, extensible et portable
- Ecrit en Java
- Permet de stocker de très gros volumes de données sur un grand nombre de machines (nœuds) équipées de disques durs banalisés → Cluster
- Quand un fichier `mydata.txt` est enregistré dans HDFS, il est décomposé en grands blocs (par défaut 64Mo), chaque bloc ayant un nom unique: `blk_1`, `blk_2`...



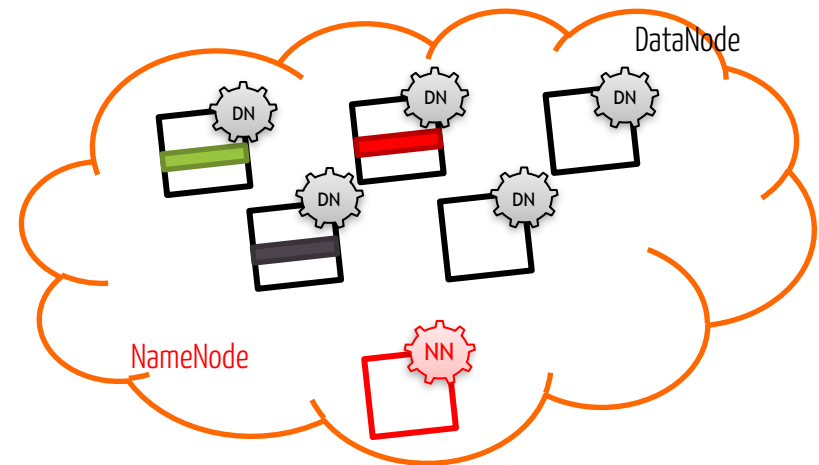
*mydata.txt* (150 Mo)



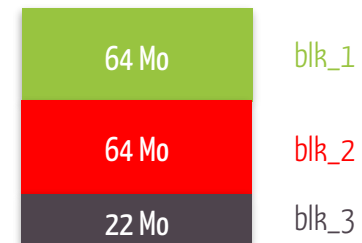
# HDFS : Hadoop Distributed File System

## Architecture

- Chaque bloc est enregistré dans un nœud différent du cluster
- DataNode : démon sur chaque nœud du cluster
- NameNode :
  - Démon s'exécutant sur une machine séparée
  - Contient des méta-données
  - Permet de retrouver les nœuds qui exécutent les blocs d'un fichier



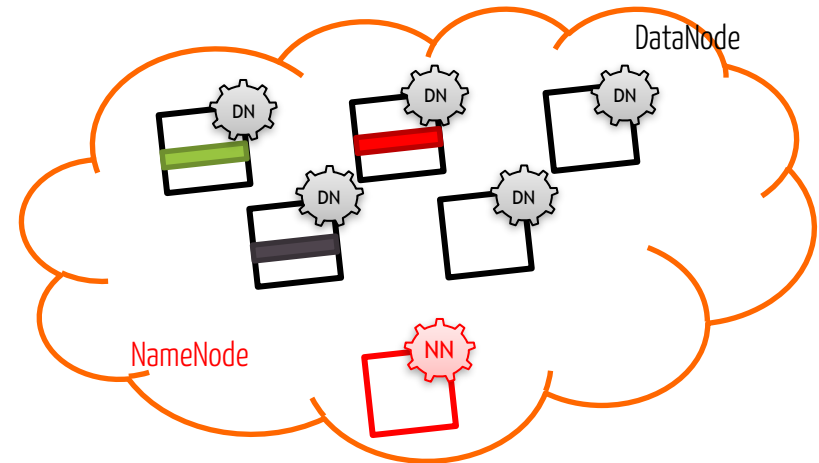
mydata.txt (150 Mo)



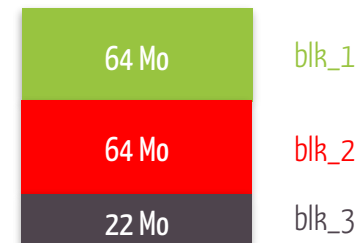
# HDFS : Hadoop Distributed File System

## Architecture

- Quels sont les problèmes possibles?
  - ✓ Panne de réseau ?
  - ✓ Panne de disque sur les DataNodes ?
  - ❑ Pas tous les DN sont utilisés ?
  - ✓ Les tailles des blocks sont différentes ?
  - ❑ Panne de disque sur les NameNodes ?



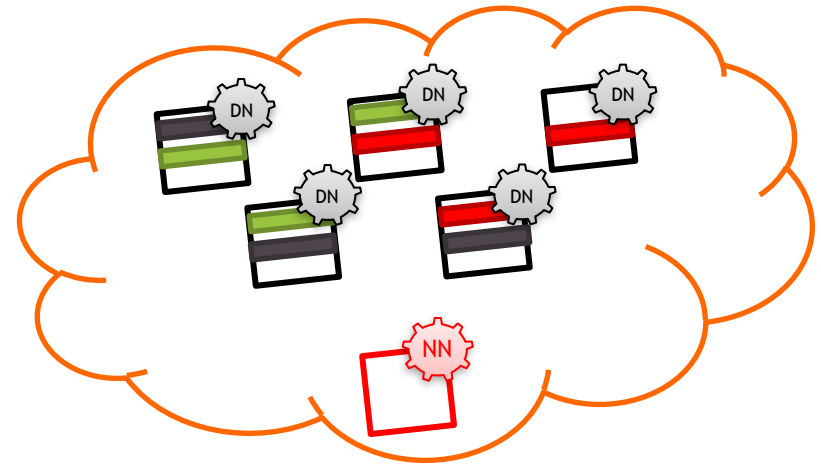
mydata.txt (150 Mo)



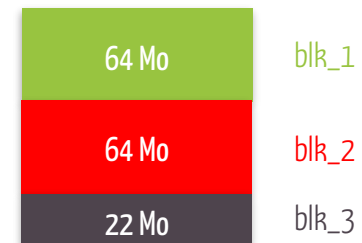
# HDFS : Hadoop Distributed File System

## Stratégie de Réplication

- Si l'un des nœuds a un problème, les données seront perdues
  - Hadoop réplique chaque bloc 3 fois (par défaut) dans chaque
  - Concept de Rack Awareness (rack = baie de stockage)
  - Il utilise la stratégie de réplication suivante:
    - Une réplique dans un nœud de la même baie que le premier
    - Une deuxième dans un nœud d'une autre baie
    - Une troisième dans un autre nœud de la même baie que le deuxième
  - Si le nœud est en panne, le NN le détecte, et s'occupe de répliquer encore les blocs qui y étaient hébergés pour avoir toujours 3 copies stockées
- Si le NameNode a un problème ?



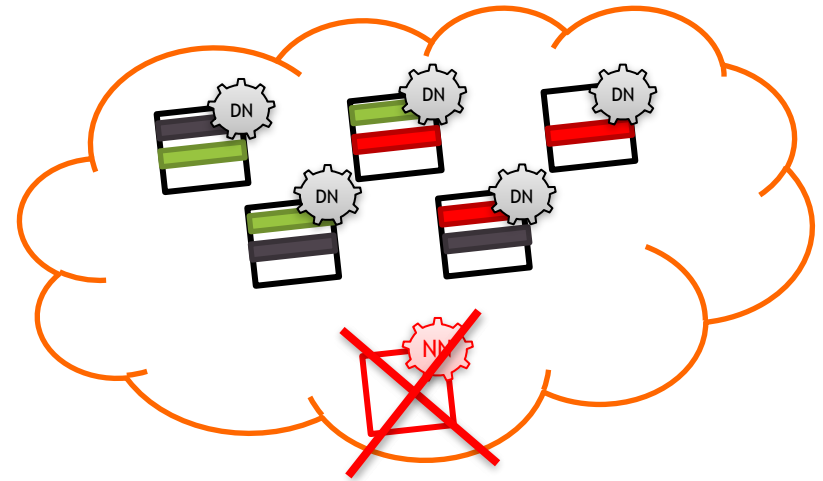
mydata.txt (150 Mo)



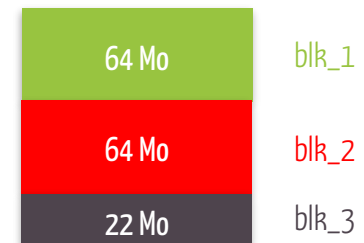
# HDFS : Hadoop Distributed File System

## NameNode

- Si le NameNode a un problème :
  - ✓ Pas de problème
  - ✓ Données perdues à jamais
  - ❑ Données inaccessibles?
- Si c'est un problème d'accès (réseau), les données sont temporairement inaccessibles
- Si le disque du NN est défaillant, les données seront perdues à jamais



mydata.txt (150 Mo)



# HDFS : Hadoop Distributed File System

## NameNode

---

- 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 persistantes :
  - FsImage
  - EditLog

# HDFS : Hadoop Distributed File System

## NameNode: FsImage et EditLog

---

- **FsImage** : Fichier qui représente un snapshot à un moment donné des métadonnées du système de fichiers
- Efficace pour la lecture, mais lourd pour des petites modifications incrémentales comme renommer un fichier
- Plutôt que de réécrire une nouvelle fsimage à chaque changement du namespace, le NameNode stocke l'opération de modification dans un **EditLog**
- Si le NN est défaillant, il est possible de restaurer son état précédent en rejouant les opérations de l'EditLog sur la FsImage pour obtenir la nouvelle FsImage
  - Opération appelée **Checkpointing**
- En plus de cela, un checkpointing régulier doit prendre place:
  - Eviter d'avoir un trop grand EditLog
  - Accélérer le démarrage de la machine



# HDFS : Hadoop Distributed File System

## NameNode: Standby and Secondary NameNodes

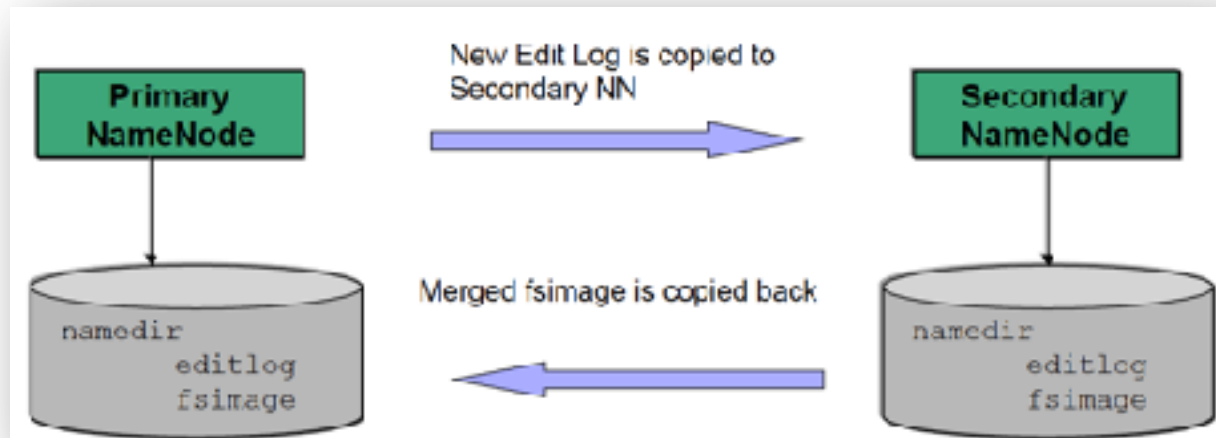
---

- La création d'une nouvelle FsImage est une opération lourde en termes de I/O et de CPU
  - Un checkpoint peut parfois prendre des minutes à s'exécuter, pendant lesquelles le NN doit restreindre l'accès des utilisateurs
- Pour décharger le NN de cette opération, HDFS la relègue à un autre noeud, appelé **Standby Namenode** (si la haute disponibilité est configurée) ou **Secondary Namenode** (sinon)
- Dans l'un ou l'autre des cas, le checkpoint est déclenché si l'une de ces deux conditions est assurée:
  - Une certaine période de temps est passée depuis le dernier checkpoint
  - Un certain nombre de transactions sont accumulées dans le EditLog

# HDFS : Hadoop Distributed File System

## Checkpoint avec un Secondary NameNode (2NN)

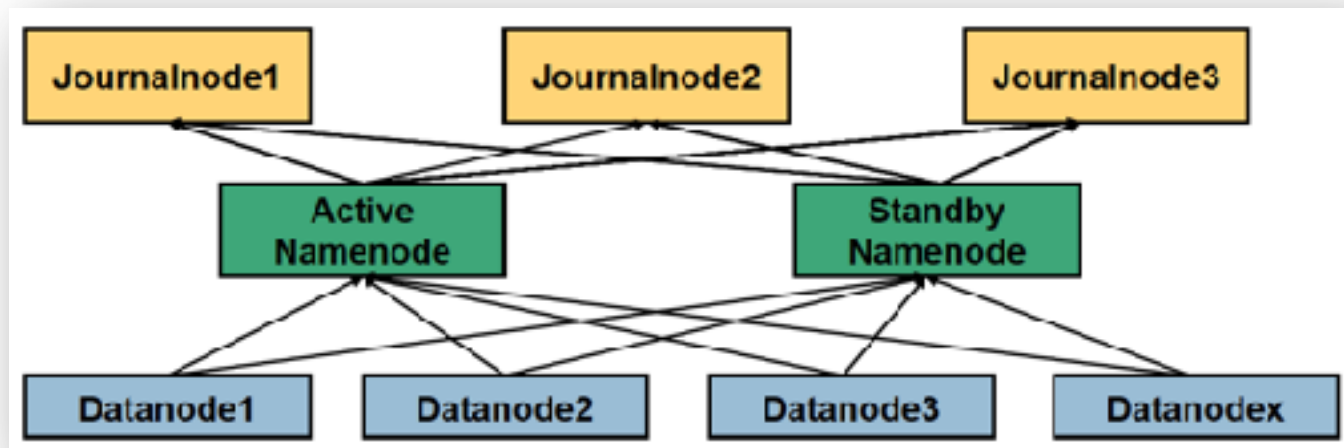
- Chaque couple de minutes, le 2NN copie l'EditLog du NN
- Il fusionne cet EditLog avec la FsImage
- Il envoie ensuite la nouvelle FsImage au NN
- Pas de haute disponibilité (HA)
  - La copie de la FsImage dans le 2NN n'est pas toujours à jour, en cas de failure du NN
- Mais
  - Moins de trafic réseau
  - Moins de données échangées
  - ...



# HDFS : Hadoop Distributed File System

## Checkpoint avec un Standby NameNode (SbNN)

- Ajoute la haute disponibilité (à partir de HDFS-2)
- Chaque modification du système de fichiers est loggée dans au moins 3 noeuds journaux par le NN
  - Le SbNN applique les changements à partir des noeuds journaux à leur arrivée
  - Si une des deux conditions (temps/nombre) est vérifiée, SbNN crée la FsImage et l'envoie au NN
- Les DataNodes envoient les emplacements des blocs et les heartbeats aux deux NameNodes
- L'état du SbNN est similaire à celui du NN
- La récupération en cas de panne est très rapide



---

Hadoop & Map Reduce

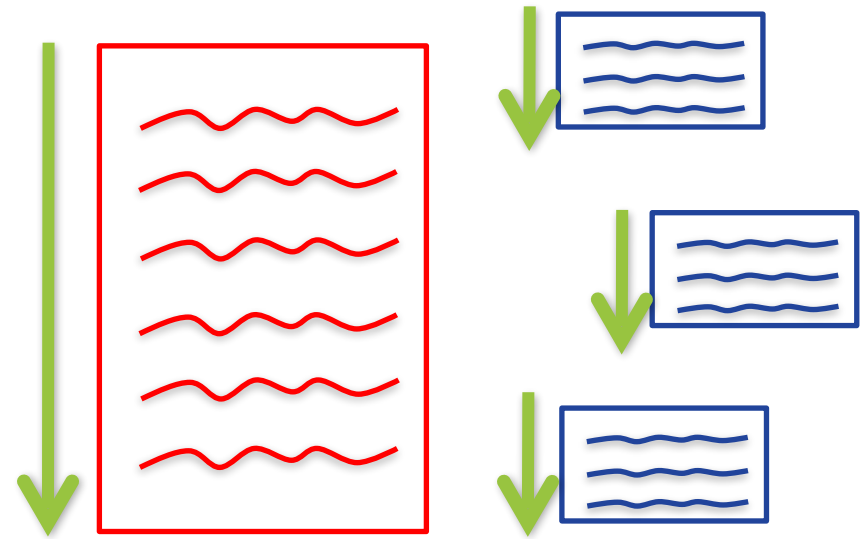
# MAP REDUCE

---

# Map-Reduce

## Définition

- Patron d'architecture de développement permettant de traiter des données volumineuses de manière parallèle et distribuée
- A la base, le langage Java est utilisé, mais grâce à une caractéristique de Hadoop appelée Hadoop Streaming, il est possible d'utiliser d'autres langages comme Python ou Ruby
- Au lieu de parcourir le fichier séquentiellement (bcp de temps), il est divisé en morceaux qui sont parcourus en parallèle.



# Map-Reduce

## Exemple

- Imaginons que vous ayez plusieurs magasins que vous gérez à travers le monde
- Un très grand livre de comptes contenant TOUTES les ventes
- Objectif : Calculer le total des ventes par magasin pour l'année en cours
- Supposons que les lignes du livres aient la forme suivante:



2012-01-01	London	Clothes	25.99
2012-01-01	Miami	Music	12.15
2012-01-02	NYC	Toys	3.10
2012-01-02	Miami	Clothes	50.00

• Jour      Ville      Produit      Prix

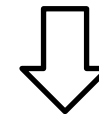
# Map-Reduce

## Exemple

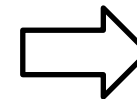
- Possibilité :
  - Pour chaque entrée, saisir la ville et le prix de vente
  - Si on trouve une entrée avec une ville déjà saisie, on les regroupe en faisant la somme des ventes
- Dans un environnement de calcul traditionnel, on utilise généralement des Hashtables, sous forme de:
  - Clef                      Valeur
- Dans notre cas, la clef serait l'adresse du magasin, et la valeur le total des ventes.



2012-01-01	London	Clothes	25.99
2012-01-01	Miami	Music	12.15
2012-01-02	NYC	Toys	3.10
2012-01-02	Miami	Clothes	50.00



London	25.99
Miami	12.15
NYC	3.10



London	25.99
Miami	62.15
NYC	3.10

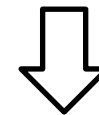
# Map-Reduce

## Exemple

- Si on utilise les hashtables sur 1To, Problèmes ?
  - ☐ Ça ne marchera pas ?
  - ☒ Problème de mémoire ?
  - ☒ Temps de traitement long ?
  - ☐ Réponses erronées ?
- Le traitement séquentiel de toutes les données peut s'avérer très long
- Plus on a de magasins, plus l'ajout des valeurs à la table est long
- Il est possible de tomber à court de mémoire pour enregistrer cette table
- Mais cela peut marcher, et le résultat sera correct



2012-01-01	London	Clothes	25.99
2012-01-01	Miami	Music	12.15
2012-01-02	NYC	Toys	3.10
2012-01-02	Miami	Clothes	50.00



Clef	Valeur
London	25.99
Miami	62.15
NYC	3.10




# Map-Reduce

## Exemple

- Map-Reduce : Moyen plus efficace et rapide de traiter ces données
- Au lieu d'avoir une seule personne qui parcourt le livre, si on en recrutait plusieurs?
- Appeler un premier groupe les Mappers et un autre les Reducers
- Diviser le livre en plusieurs parties, et en donner une à chaque Mapper
  - Les Mappers peuvent travailler en même temps, chacun sur une partie des données

Mappers



2012-01-01	London	City Hall	25.99
2012-01-01	Miami	Music	12.15
2012-01-01	NYC	Trav	1.91
2012-01-02	Miami	City Hall	30.00

2012-01-01	London	City Hall	25.99
2012-01-01	Miami	Music	12.15
2012-01-01	NYC	Trav	1.91
2012-01-02	Miami	City Hall	30.00

2012-01-01	London	City Hall	25.99
2012-01-01	Miami	Music	12.15
2012-01-01	NYC	Trav	1.91
2012-01-02	Miami	City Hall	30.00



Reducers

# Map-Reduce

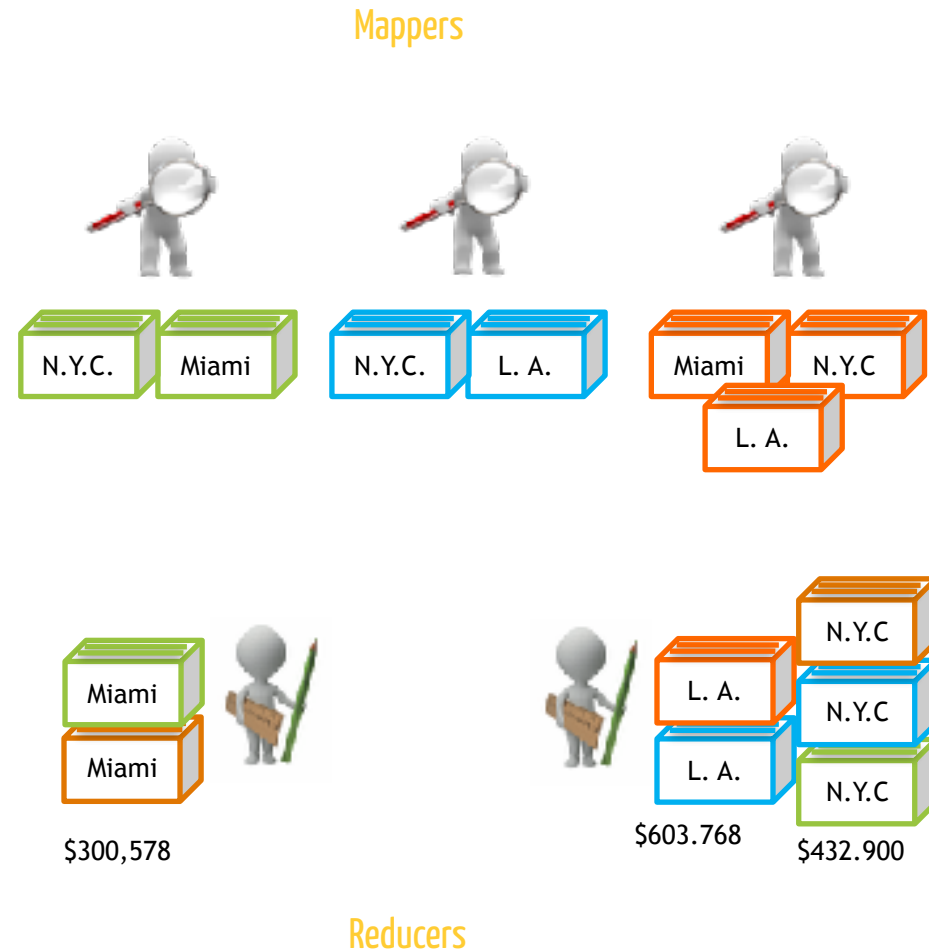
## Exemple

### Mappers

- Pour chaque entrée, saisir la ville, et le total des ventes et les enregistrer dans une fiche
- Rassembler les fiches du même magasin dans une même pile

### Reducers

- Chaque Reducer sera responsable d'un ensemble de magasins
- Ils collectent les fiches qui leur sont associées des différents Mappers
- Ils regroupent les petites piles d'une même ville en une seule
- Ils parcourent ensuite chaque pile par ordre alphabétique des villes (L.A avant Miami), et font la somme de l'ensemble des enregistrements



# Map-Reduce

## Exemple

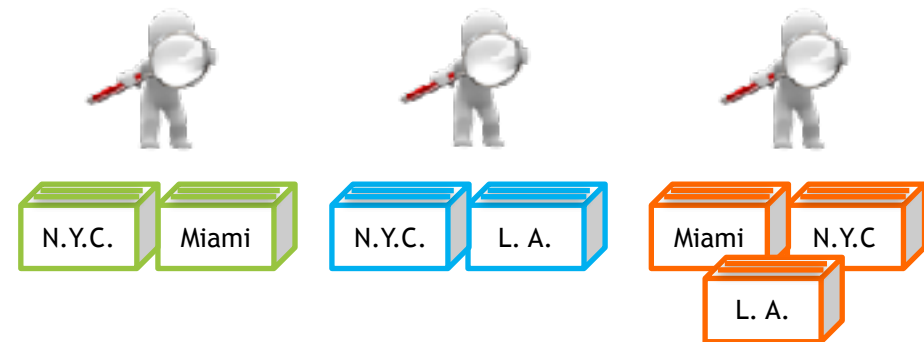
- Le Reducer reçoit des données comme suit:

- Miami 12.34
- Miami 99.07
- Miami 3.14
- NYC 99.77
- NYC 88.99

- Pour chaque entrée, de quoi avons-nous besoin pour calculer la totalité des ventes pour chaque magasin?

- ☐ Coût précédent
- ☒ Coût en cours
- ☐ Ventes totales par magasin
- ☒ Magasin précédent
- ☐ Magasin en cours

Mappers

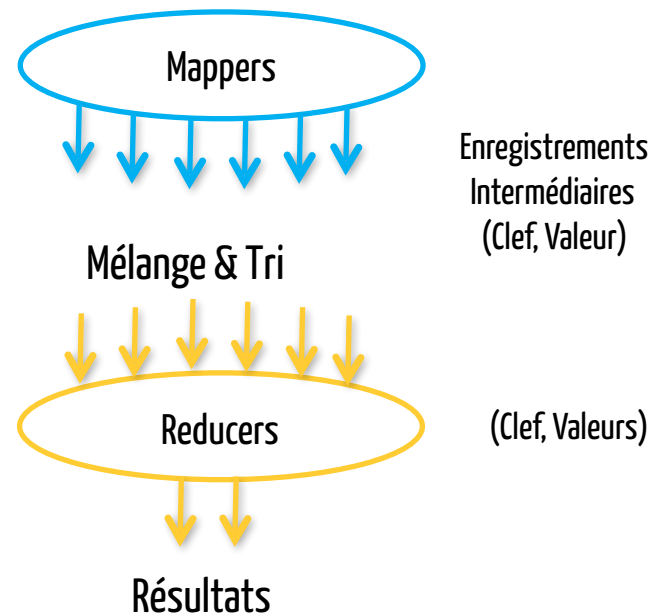


Reducers

# Map-Reduce

## Fonctionnement

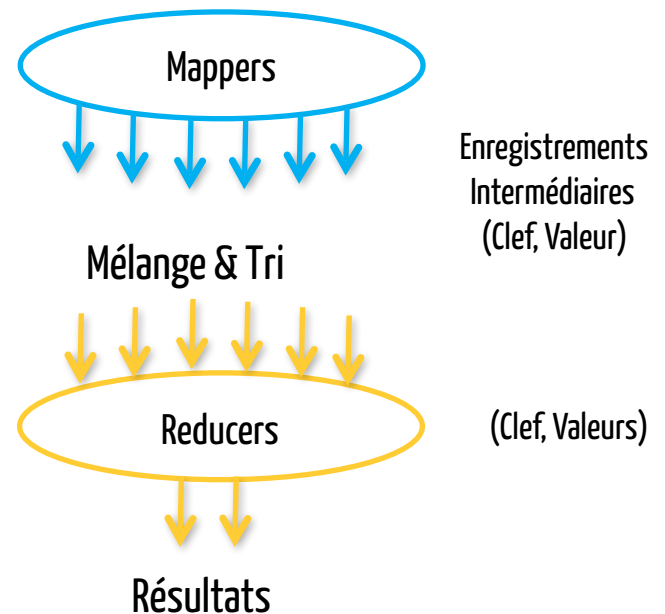
- Les Mappers sont des petits programmes qui commencent par traiter chacun une petite partie des données
- Ils fonctionnent en parallèle
- Leurs sorties représentent les enregistrements intermédiaires: sous forme d'un couple (clef, valeur)
- Une étape de Mélange et Tri s'ensuit
  - Mélange : Sélection des piles de fiches à partir des Mappers
  - Tri : Rangement des piles par ordre au niveau de chaque Reducer
- Chaque Reducer traite un ensemble d'enregistrements à la fois, pour générer les résultats finaux



# Map-Reduce

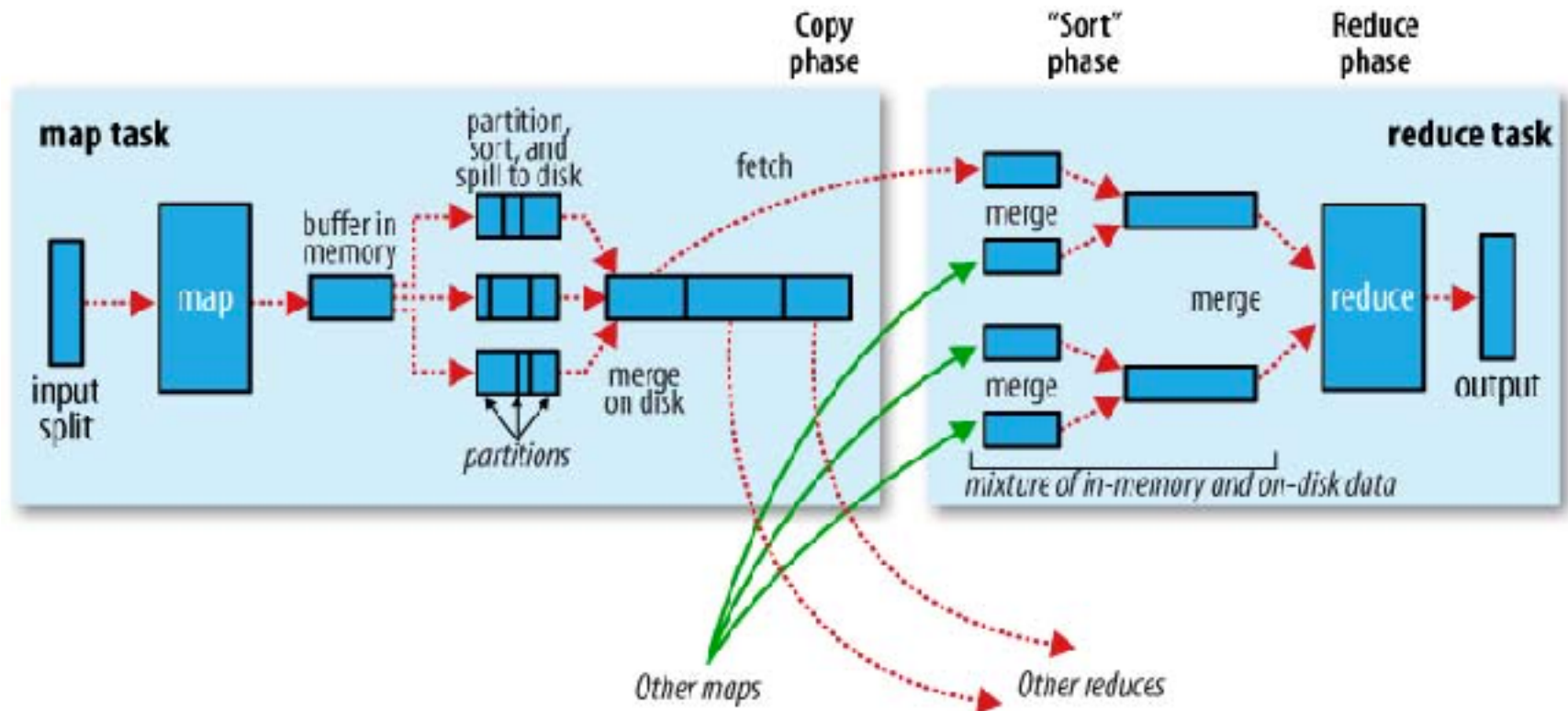
## Résultats

- Pour avoir un résultat trié par ordre, on doit:
  - Soit avoir un seul Reducer, mais ça ne se met pas bien à l'échelle
  - Soit ajouter une autre étape permettant de faire le tri final
- Si on a plusieurs Reducers, on ne peut pas savoir lesquels traitent quelles clefs: le partitionnement est aléatoire.



# Map-Reduce

## Phases

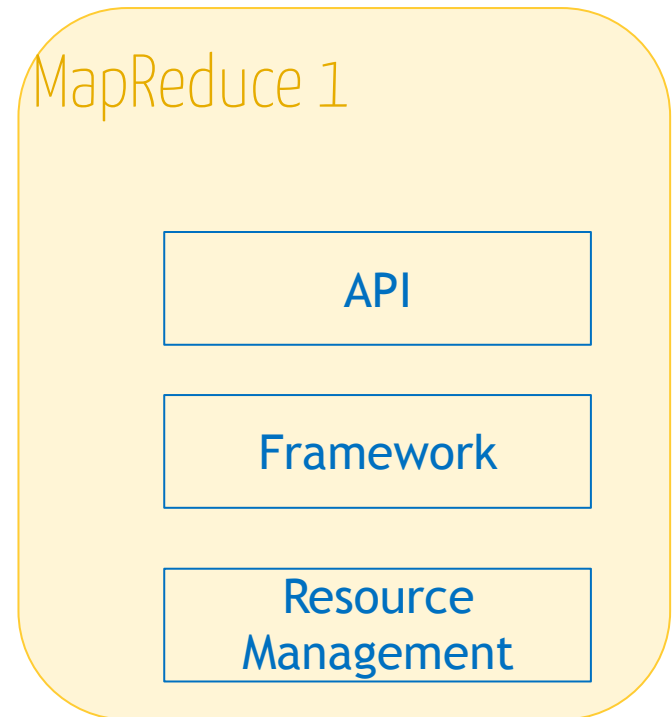


# MapReduce V1 (MRv1)

## Composants

MapReduce v1 intègre trois composants

- API
  - Pour permettre au programmeur l'écriture d'applications MapReduce
- Framework
  - Services permettant l'exécution des Jobs MapReduce, le Shuffle/Sort...
- Resource Management
  - Infrastructure pour gérer les noeuds du cluster, allouer des ressources et ordonnancer les jobs



# MapReduce V1 (MRv1)

## Démons

---

- JobTracker
  - Divise le travail sur les Mappers et Reducers, s'exécutant sur les différents nœuds
- TaskTracker
  - S'exécute sur chacun des nœuds pour exécuter les vraies tâches de Map-Reduce
  - Choisit en général de traiter (Map ou Reduce) un bloc sur la même machine que lui
  - S'il est déjà occupé, la tâche revient à un autre tracker, qui utilisera le réseau (rare)



# MapReduce V1 (MRv1)

## Fonctionnement

---

- Un job Map-Reduce (ou une Application Map-Reduce) est divisé en plusieurs tâches appelées mappers et reducers
- Chaque tâche est exécutée sur un noeud du cluster
- Chaque noeud a un certain nombre de slots prédéfinis:
  - Map Slots
  - Reduce Slots
- Un slot est une unité d'exécution qui représente la capacité du task tracker à exécuter une tâche (map ou reduce) individuellement, à un moment donné
- Le Job Tracker se charge à la fois:
  - D'allouer les ressources (mémoire, CPU...) aux différentes tâches
  - De coordonner l'exécution des jobs Map-Reduce
  - De réserver et ordonnancer les slots, et de gérer les fautes en réallouant les slots au besoin

# MapReduce V1 (MRv1)

## Problèmes

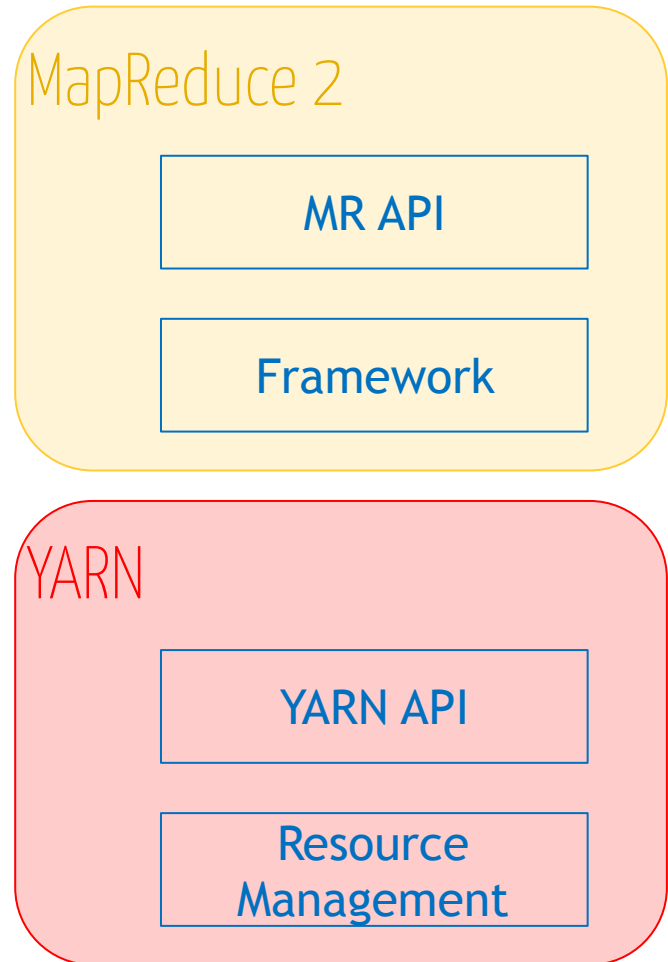
---

- Le Job Tracker s'exécute sur une seule machine, et fait plusieurs tâches (gestion de ressources, ordonnancement et monitoring des tâches...)
  - Problème de scalabilité: les nombreux datanodes existants ne sont pas exploités, et le nombre de noeuds par cluster limité à 4000
- Si le Job Tracker tombe en panne, tous les jobs doivent redémarrer
  - Problème de disponibilité: SPoF
- Le nombre de map slots et de reduce slots est prédéfini
  - Problème d'exploitation: si on a plusieurs map jobs à exécuter, et que les map slots sont pleins, les reduce slots ne peuvent pas être utilisés, et vice-versa
- Le Job Tracker est fortement intégré à Map Reduce
  - Problème d'interopérabilité: impossible d'exécuter des applications non-MapReduce sur HDFS

# MapReduce V2 (MRv2)

## Composants

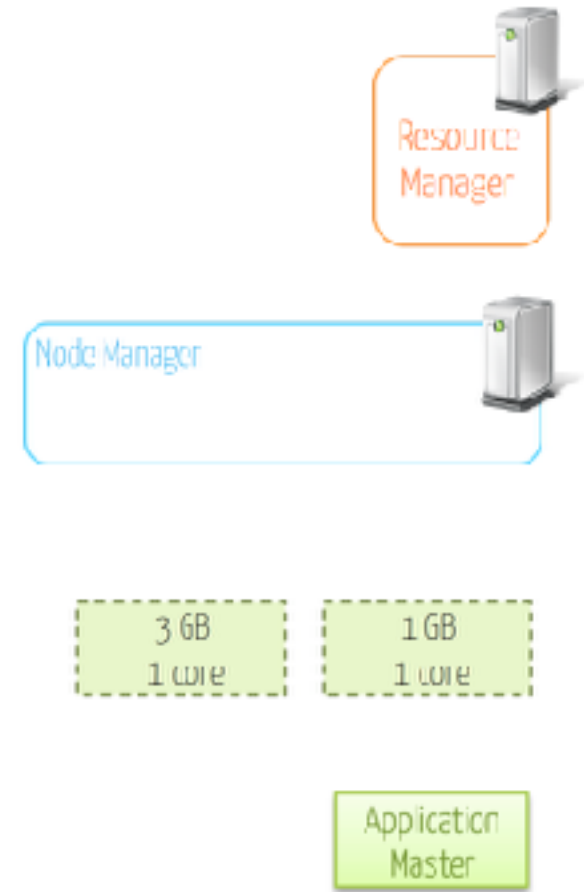
- MapReduce v2 sépare la gestion des ressources de celle des tâches MR
- Pas de notion de slots:
  - Les noeuds ont des ressources (CPU, mémoire..) allouées aux applications à la demande
- Définition de nouveaux démons
  - La plupart des fonctionnalités du Job Tracker sont déplacées vers le Application Master
  - Un cluster peut avoir plusieurs Application Masters
- Support les applications MR et non-MR



# MapReduce V2 (MRv2)

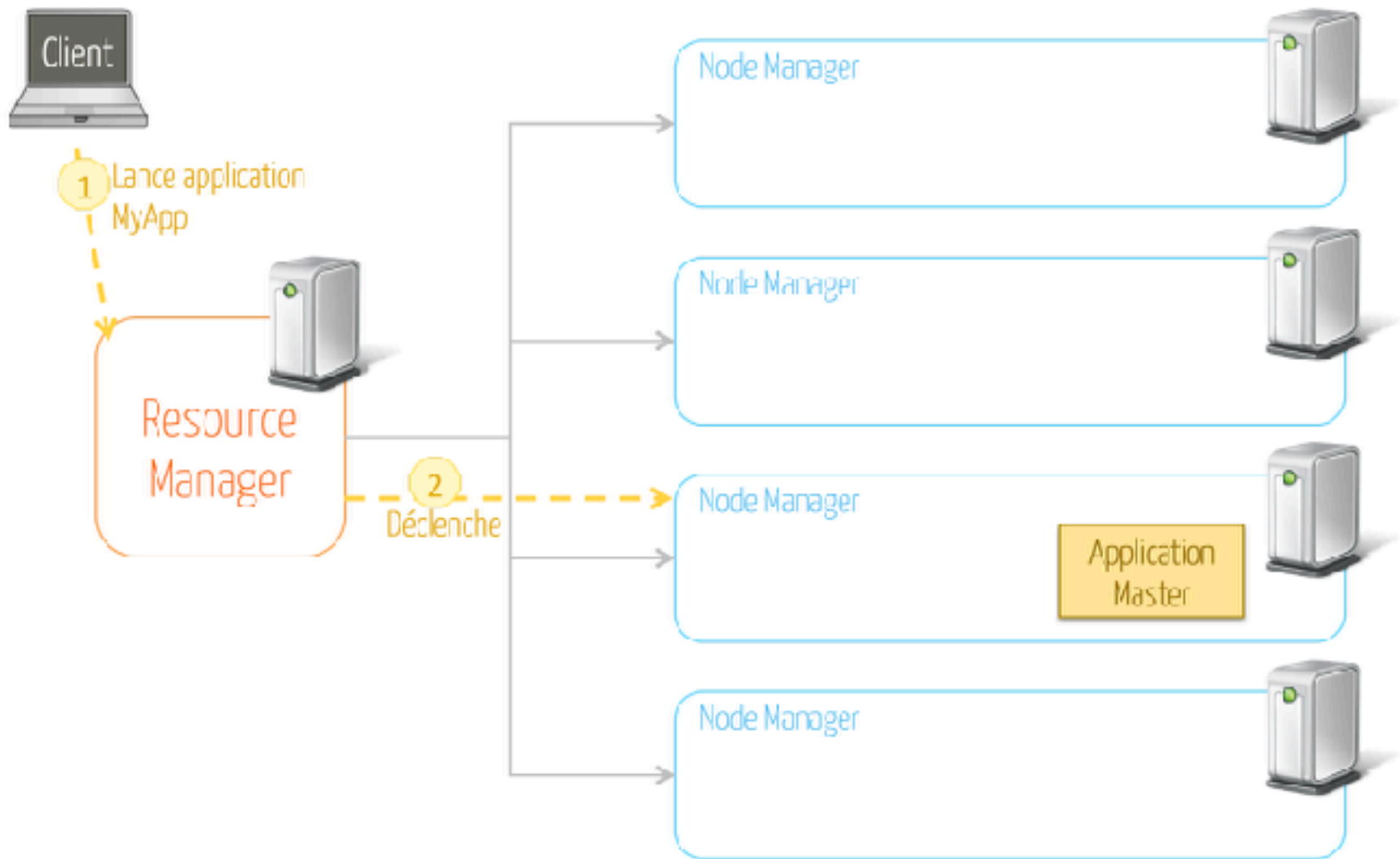
## Démons

- Resource Manager (RM)
  - Tourne sur le noeud master
  - Ordonnanceur global des ressources
  - Permet l'arbitrage des ressources entre plusieurs applications
- Node Manager (NM)
  - S'exécute sur les noeuds esclaves
  - Communique avec RM
- Containers
  - Créés par RM à la demande
  - Se voit allouer des ressources sur le noeud esclave
- Application Master (AM)
  - Un seul par application
  - S'exécute sur un container
  - Demande plusieurs containers pour exécuter les tâches de l'application



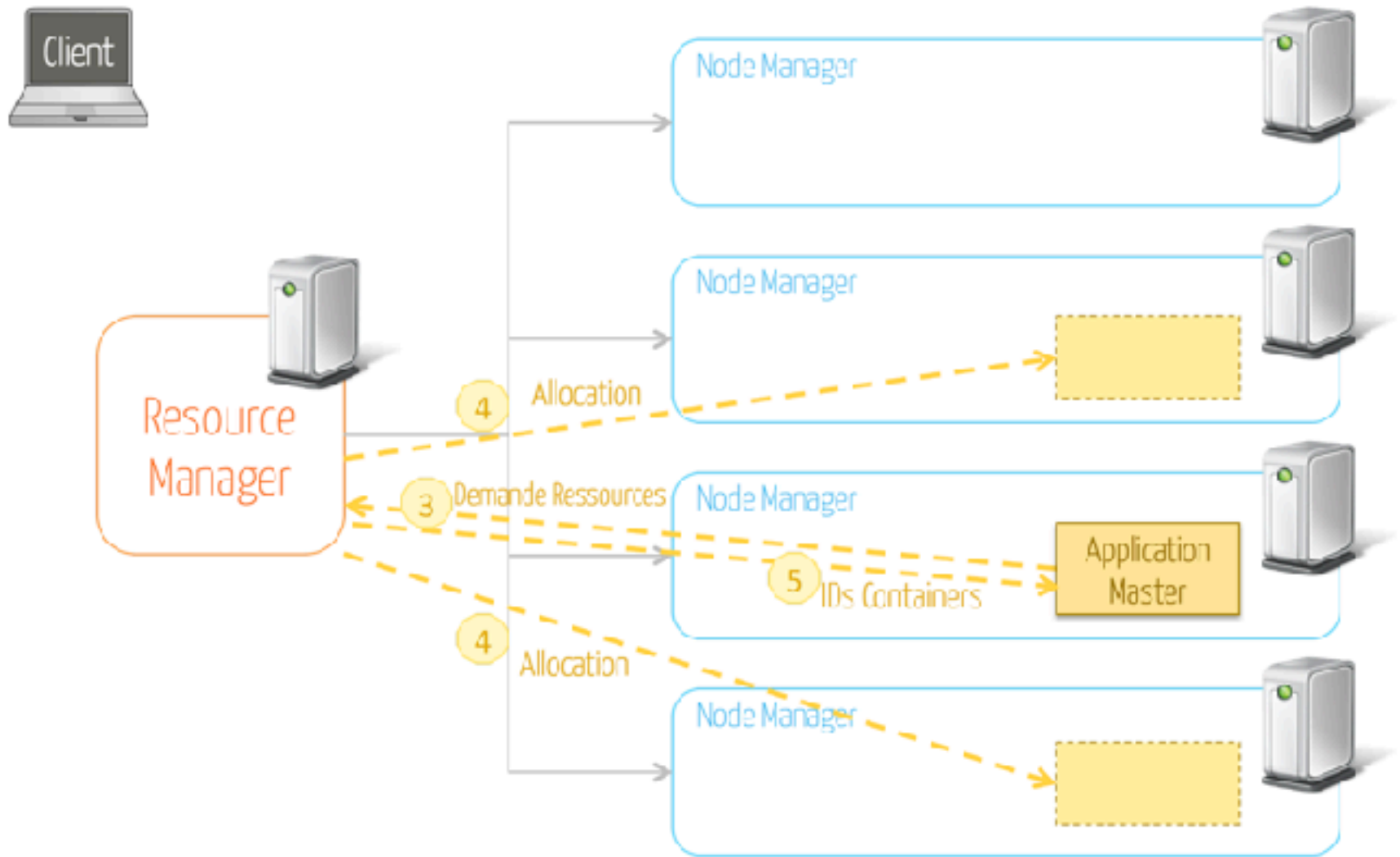
# MapReduce V2 (MRv2)

## Lancement d'une Application dans un Cluster YARN



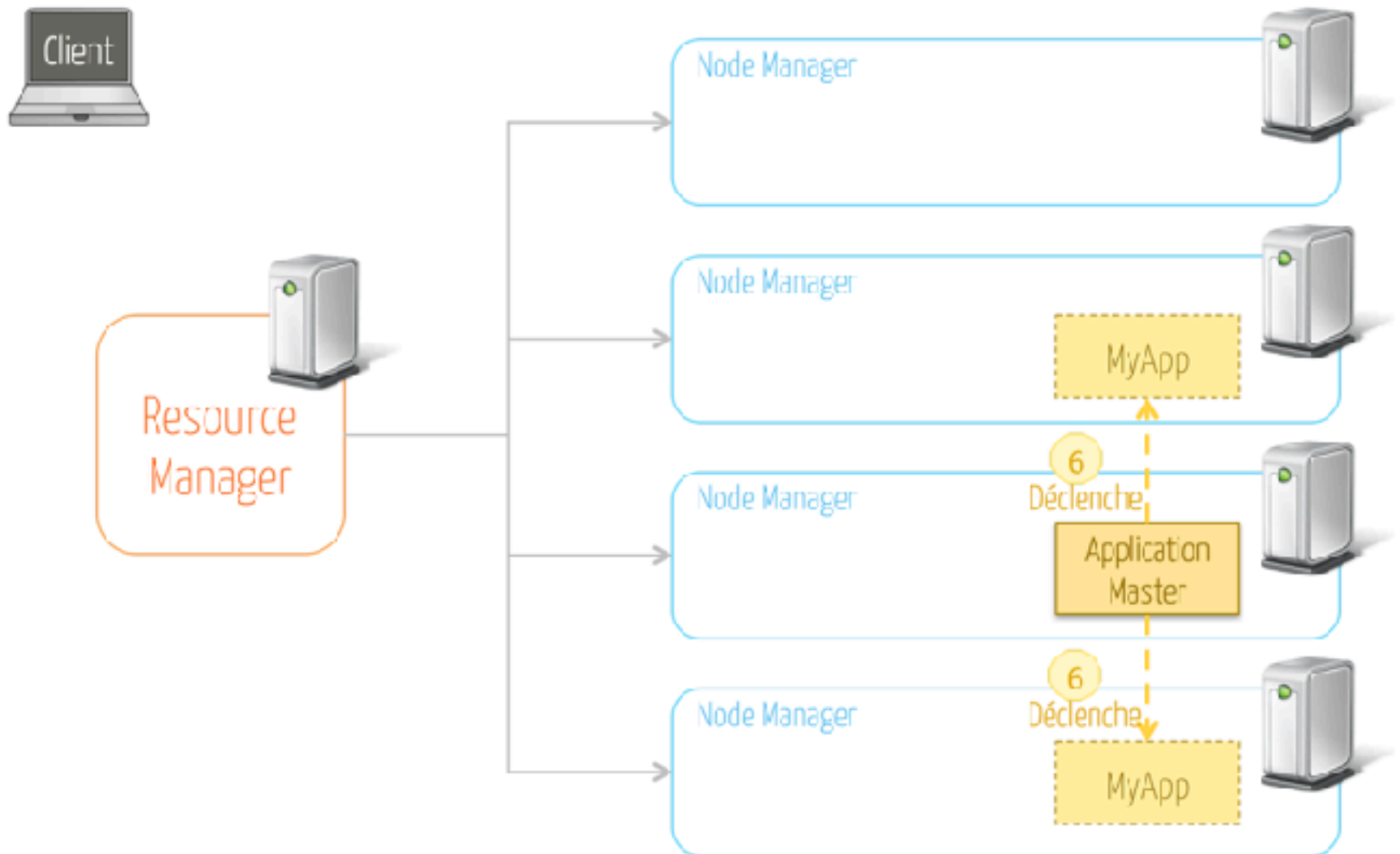
# MapReduce V2 (MRv2)

## Lancement d'une Application dans un Cluster YARN



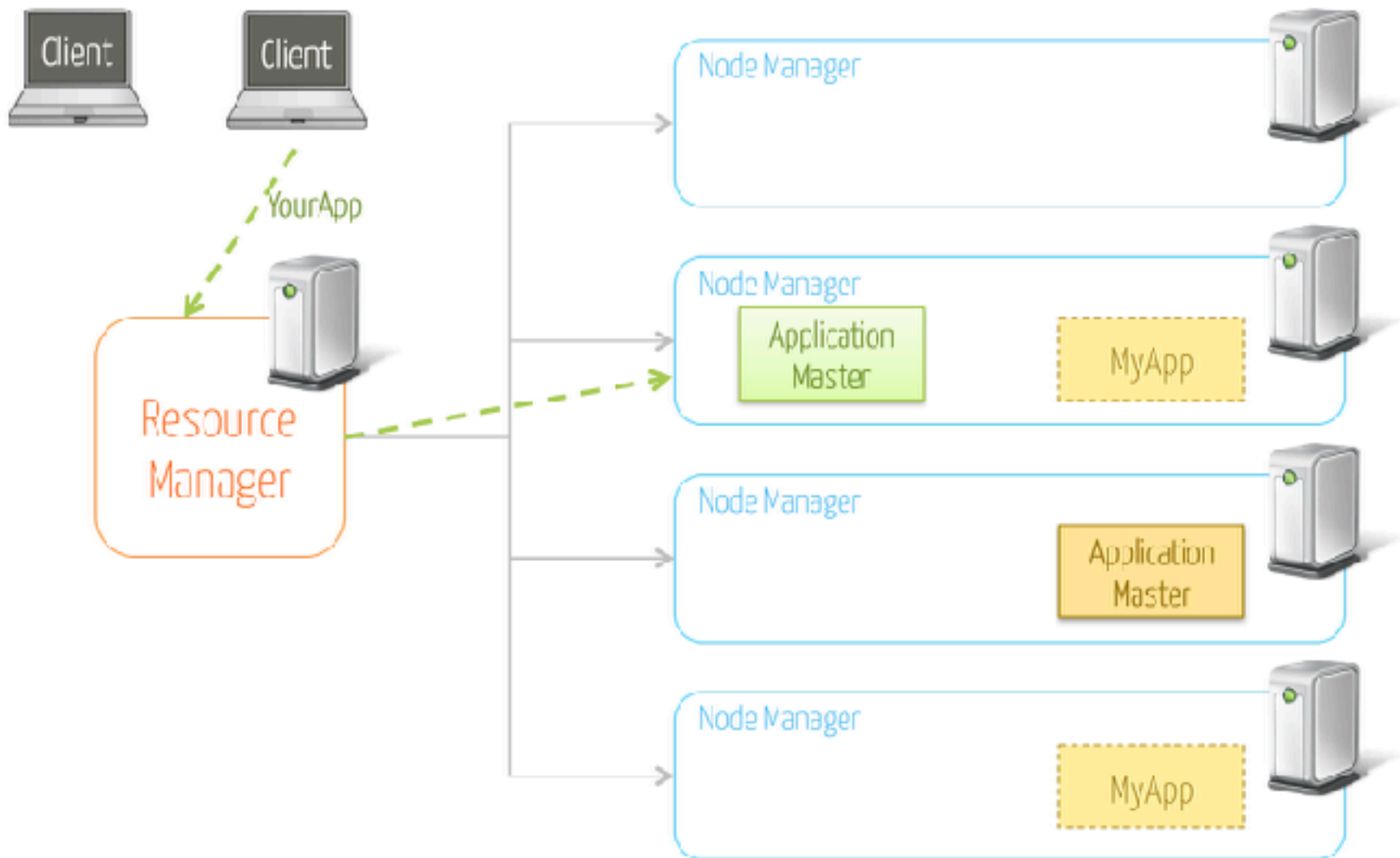
# MapReduce V2 (MRv2)

## Lancement d'une Application dans un Cluster YARN



# MapReduce V2 (MRv2)

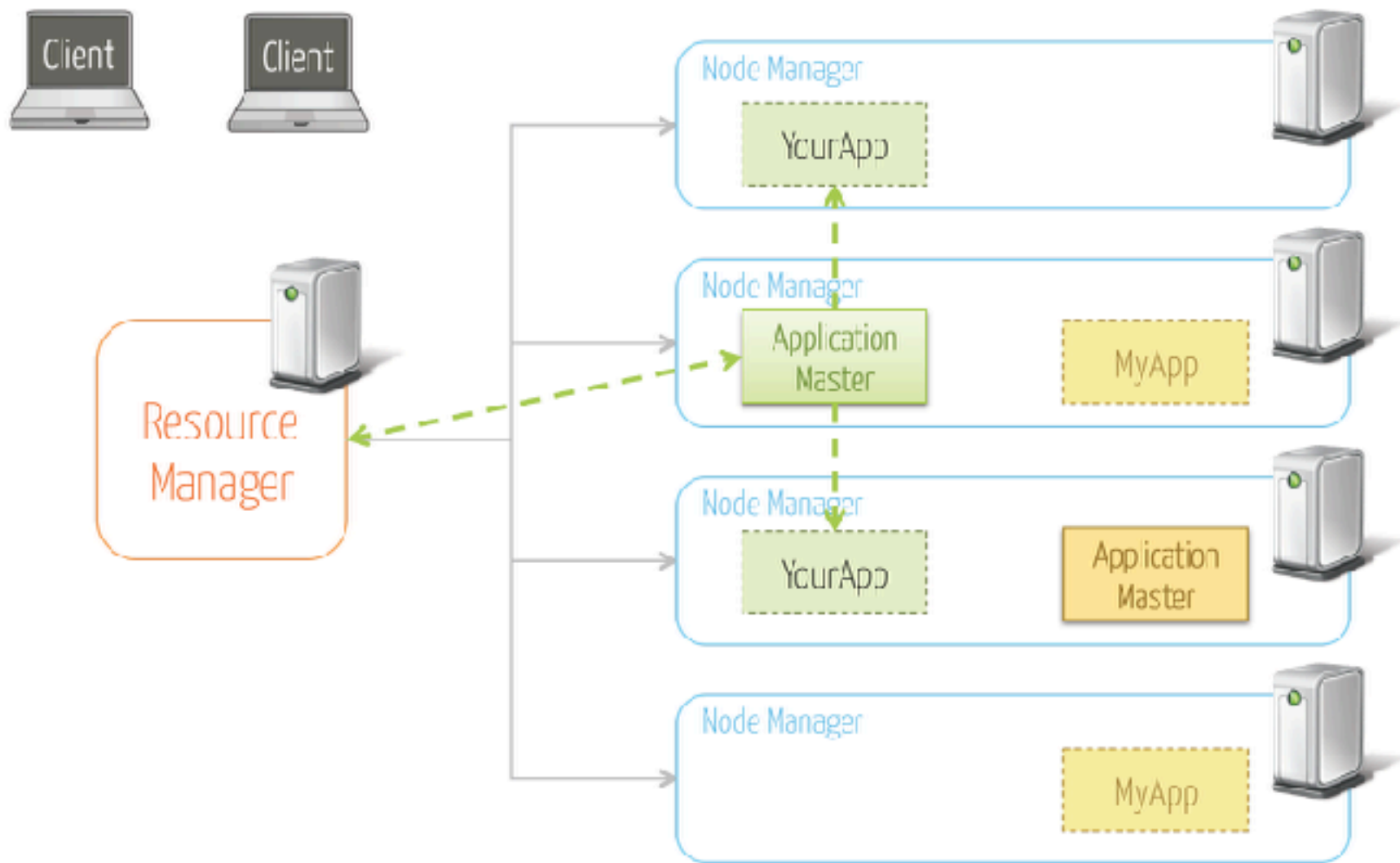
## Lancement d'une Application dans un Cluster YARN





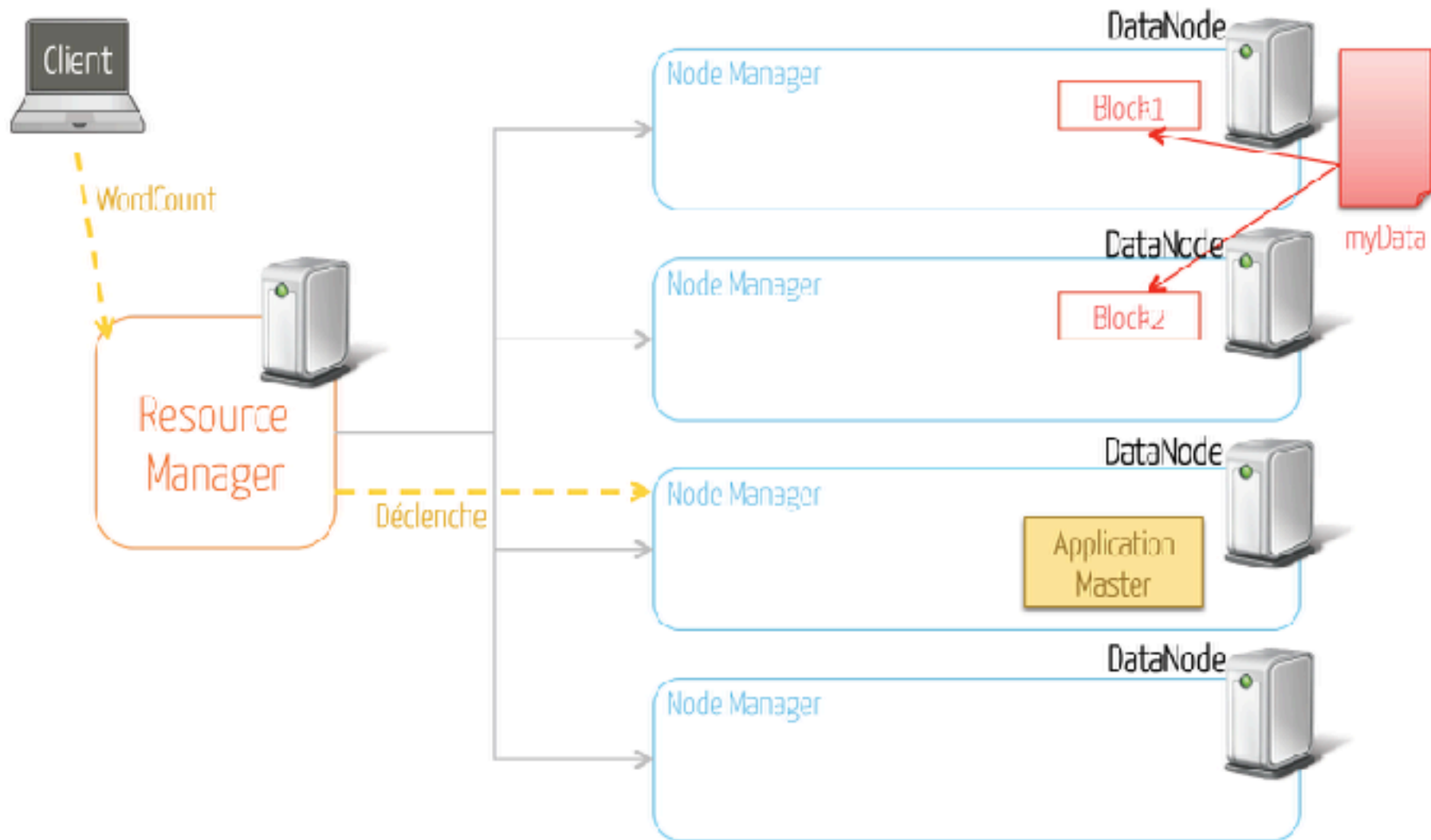
# MapReduce V2 (MRv2)

## Lancement d'une Application dans un Cluster YARN



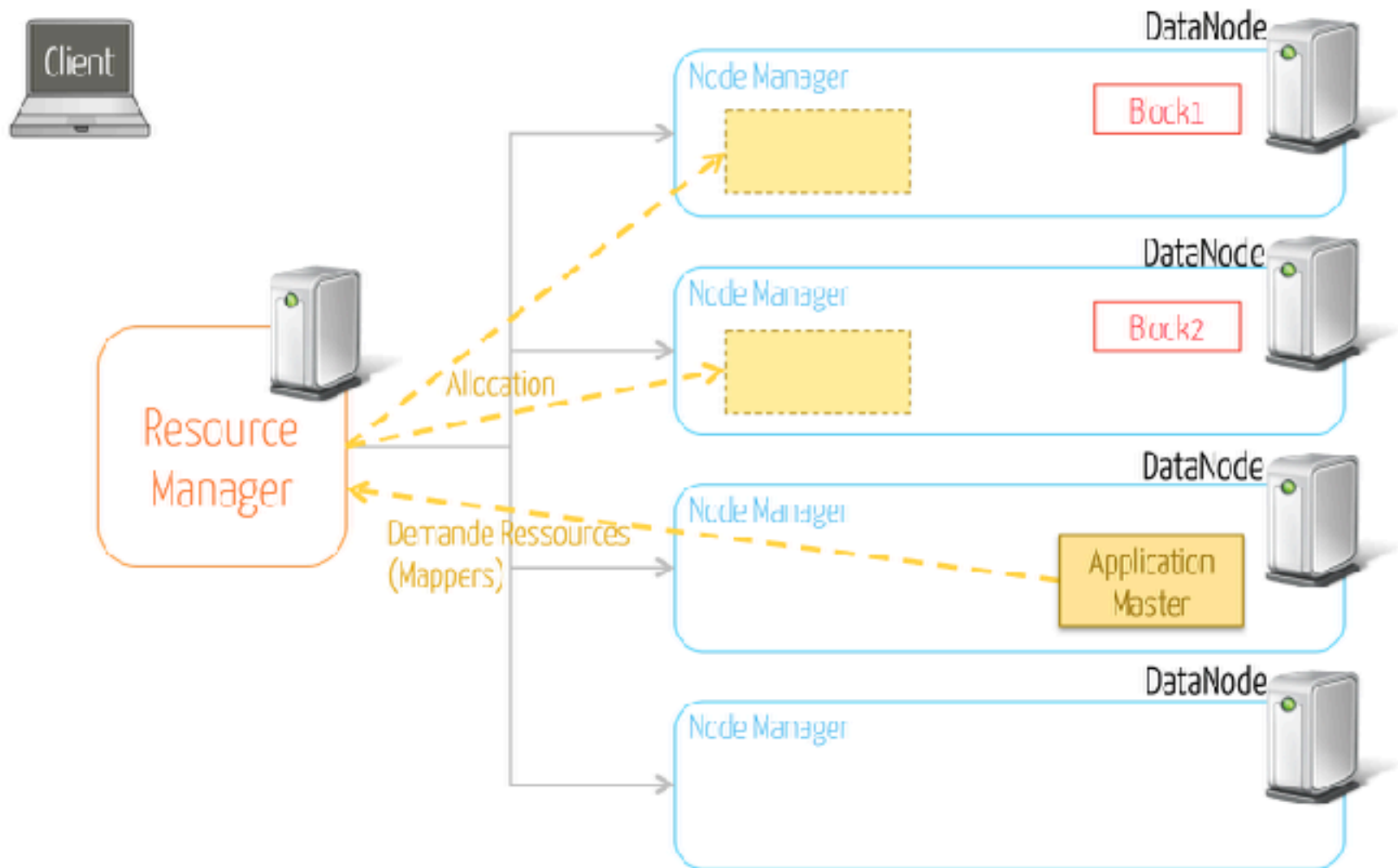
# MapReduce V2 (MRv2)

## Exécution d'un Job Map Reduce



# MapReduce V2 (MRv2)

## Exécution d'un Job Map Reduce

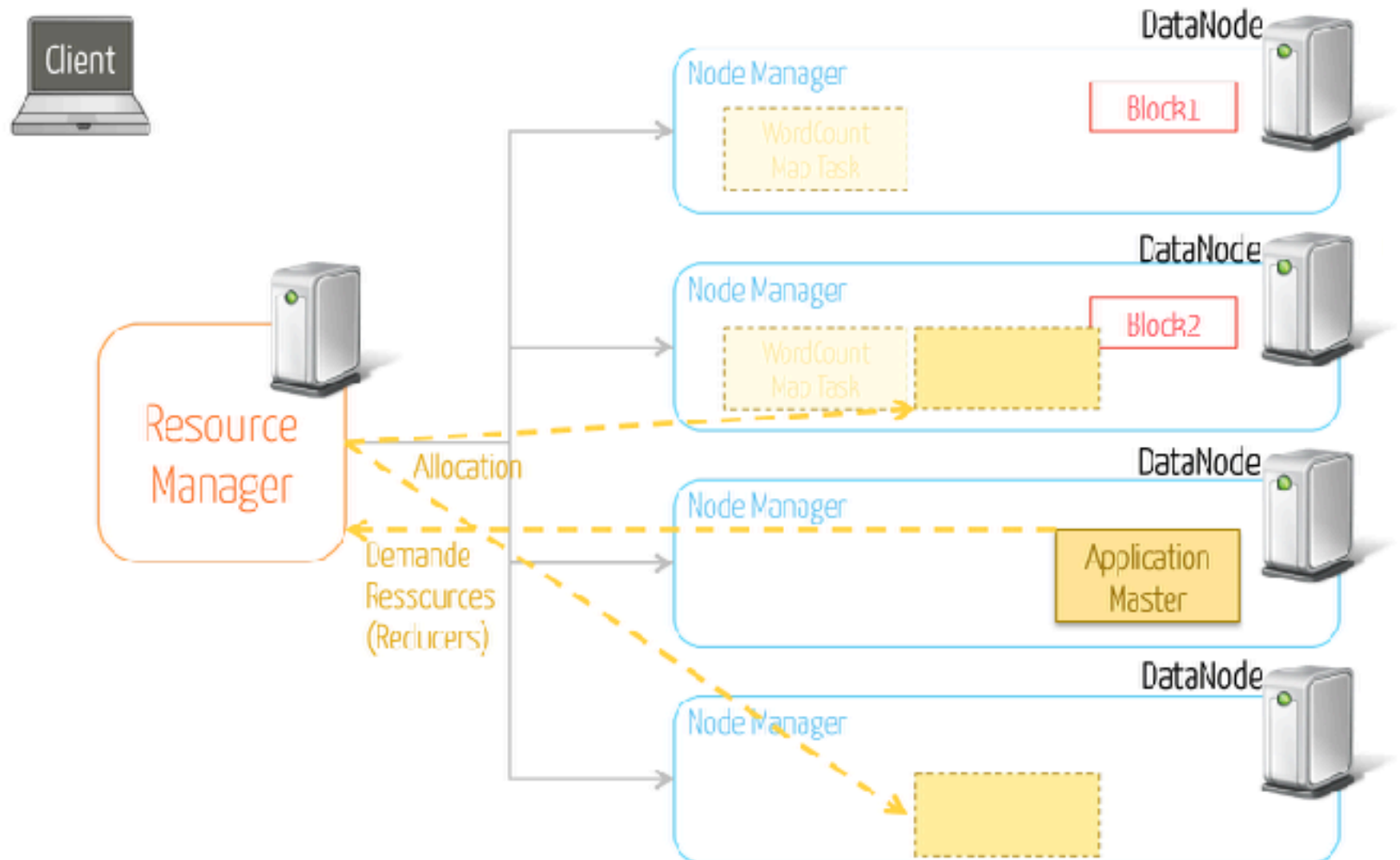


## Exécution d'un Job Map Reduce



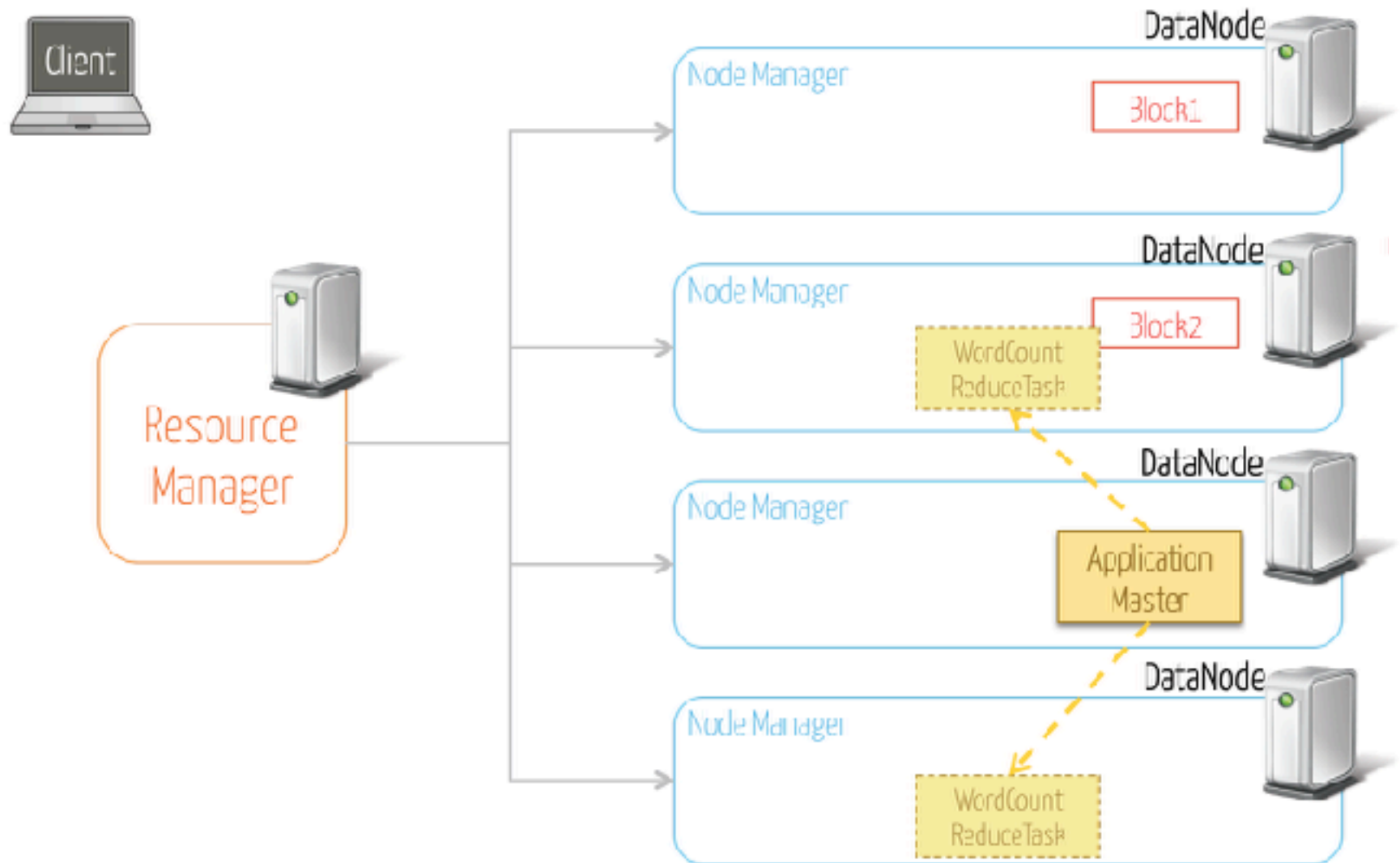
# MapReduce V2 (MRv2)

## Exécution d'un Job Map Reduce



# MapReduce V2 (MRv2)

## Exécution d'un Job Map Reduce



---

Hadoop & Map Reduce

# MAP REDUCE DESIGN PATTERNS

---

# Hadoop Design Patterns

## Présentation

---

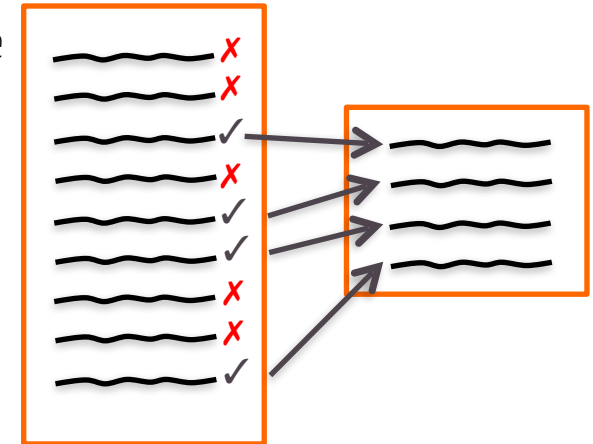
- Les designs patterns (Patrons de Conception) représentent les types de traitements les plus utilisés avec Hadoop
- Classés en trois catégories:
  - Patrons de Filtrage (Filtering Patterns)
    - Echantillonnage de données
    - Listes des top-n
  - Patrons de Récapitulation (Summarization Patterns)
    - Comptage des enregistrements
    - Trouver les min et les max
    - Statistiques
    - Indexes
  - Patrons Structurels
    - Combinaison de données relationnelles



# Patrons de Filtrage

## Présentation

- Ne modifient pas les données
- Trient, parmi les données présentes, lesquelles garder et lesquelles enlever
- On peut obtenir:
  - Des filtres simples : définition d'une fonction indiquant le critère de filtrage
  - L'échantillonnage (sampling): création d'un petit ensemble d'enregistrements à partir d'un grand ensemble, en retenant des échantillons (comme la valeur la plus élevée d'un champs particulier)
  - L'échantillonnage aléatoire : retenir un échantillon représentatif des données initiales
  - Les listes Top-n



# Patrons de Filtrage

## Présentation

---

- Exemple de Filtrage Simple:
  - Cas d'étude : fichier contenant tous les posts des utilisateurs sur un forum
  - Filtre : Retenir les posts les plus courts, contenant une seule phrase
    - Une phrase est un post qui ne contient aucune ponctuation de la forme: .!?, ou alors une seule à la fin.

```
def mapper():  
    reader = csv.reader(sys.stdin, delimiter='\t')  
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='\"'  
                        , quoting=csv.QUOTE_ALL)  
  
    for line in reader:  
  
        for i in line:  
            #print('-',i)  
            if len(i) == 0:  
                continue  
            if "!" in i[:-1]:  
                continue  
            if "." in i[:-1]:  
                continue  
            if "?" in i[:-1]:  
                continue  
            else:  
                writer.writerow(line)
```

# Patrons de Filtrage

## Présentation

---

- Exemple: Top 10
  - Trouver parmi les différents posts des forums, les 10 posts les plus longs
  - Dans une Base de données Relationnelle:
    - Trier les données
    - Extraire les 10 premières
  - Map-Reduce (de la même manière qu'une sélection sportive)
    - Chaque Mapper génère une liste Top-10
    - Le Reducer trouve les Top 10 globaux

```
def mapper():  
    a = []  
    b = []  
    reader = csv.reader(sys.stdin, delimiter='\t')  
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='\"',  
                        quoting=csv.QUOTE_ALL)  
  
    for line in reader:  
        for i in line:  
            if len(i) == 0 :  
                continue  
            else:  
                a.append(line)  
  
    a.sort(key=lambda a: (int)(a[4]), reverse=True)  
  
    for i in range(0,10):  
        b.append(a[i])  
    b.sort(key=lambda b: (int)(b[4]))  
  
    for b1 in b:  
        writer.writerow(b1)
```

# Patrons de Récapitulation

## Présentation

---

- Permettent de vous donner une idée de haut niveau de vos données
- On distingue deux types:
  - Index : Tels que les index à la fin d'un livre, ou les index utilisés par google pour représenter les pages web
  - Récapitulation (ou résumé) numérique : par exemple:
    - Chercher des chiffres, des comptes (combien dispose-t-on d'un certain type d'entrées)
    - Min et Max
    - Premier et dernier
    - Moyenne
    - ...

# Patrons de Récapitulation

## Index

---

- Les index permettent une recherche plus rapide
- Dans un livre: pour chaque mot donné, indiquer les différentes pages où se trouve ce mot
- Dans le web: on trouve des liens vers des pages web à partir d'un ensemble de mots clefs

# Patrons de Récapitulation

## Index

- Exemple : Indexation des mots dans les posts d'un forum

- Mapper →

```
import sys
import csv
import re

firstLine = 1

reader = csv.reader(sys.stdin, delimiter='\t')
writer = csv.writer(sys.stdout, delimiter='\t', quotechar='"', quoting=csv.QUOTE_ALL)

for line in reader:
    if firstLine == 1:
        #Si on se trouve dans la premiere ligne (celle des titres), sauter c
        #ette ligne
        firstLine = 0
        continue
    body = line[4]
    node = line[0]
    words = re.findall(r"[\w']+|[.,!?:;]", body)
    for word in words:
        if word not in ('.', '!', ',', '?', '#', '$', '[', ']', '/', '\'', '<', '>', '=',
            '- ', ':', ';', '(', ')'):
            print "{0}\t{1}".format(word, node)
```

# Patrons de Récapitulation

## Index

- Exemple : Indexation des mots dans les posts d'un forum

- Reducer →

```
import sys

nbTotal = 0
oldWord = None
listNodes = []

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisWord, thisNode = data_mapped

    if oldWord and oldWord.lower() != thisWord.lower():
        listNodes.sort(key=lambda listNodes: (int)(listNodes))
        print oldWord, "\t", nbTotal, "\t", listNodes
        oldWord = thisWord.lower()
        nbTotal = 0
        listNodes = []

    oldWord = thisWord.lower()
    nbTotal = nbTotal + 1
    if thisNode not in listNodes:
        listNodes.append(thisNode)

if oldWord != None:
    listNodes.sort(key=lambda listNodes: (int)(listNodes))
```

# Patrons de Récapitulation

## Récapitulations Numériques

---

- Peuvent être:
  - Le nombre de mots, enregistrements...
    - Souvent: la clef = l'objet à compter, et la valeur = 1 (nombre d'occurrences)
  - Min-Max / Premier-Dernier
  - La moyenne
  - La médiane
  - Écart type
  - ...
- Exemple de question:
  - Y'a-t-il une relation entre le jour de la semaine et la somme dépensée par les clients?
  - Mapper : clef = jour\_de\_la\_semaine;    valeur = somme\_dépensée
  - Reducer : calcul



# Patrons de Récapitulation

## Mélangeur

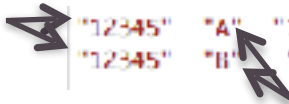
---

- Possibilité d'utiliser un mélangeur (Combiner) entre les mappers et les reducers
- Permettent de réaliser la réduction en local dans chacun des DataNodes Mappers AVANT de faire appel au nœud réducteur principal.
- Exemple: pour calculer la moyenne des ventes par jour de la semaine:
  - Sans Combiner
    - Les Mappers parcourent les données, et affichent le couple (Jour\_de\_la\_semaine, montant)
    - Pour chaque jour, le Reducer conserve une somme et un compteur
    - Il divise à la fin la somme par le compteur
  - Avec Combiner
    - Chaque nœud réalise une première réduction où les moyennes locales sont calculées
    - Le Reducer final regroupe ces moyennes et synthétise la moyenne finale
    - Nombre d'enregistrements envoyés au réducteur significativement réduit
    - Temps nécessaire pour la réduction diminue

# Patrons Structurels

## Présentation

- Utilisés quand les données proviennent d'une base de données structurée
- Plusieurs tables, donc plusieurs sources de données, liées par clef étrangère
- Les données des différentes tables sont exportées sous forme de fichiers délimités
- Le Mapper aura donc comme tâche de :
  - Parcourir l'ensemble des fichiers correspondant aux tables de la base
  - Extraire de chacune des entrées les données nécessaires, en utilisant comme clef la clef étrangère joignant les deux tables
  - Afficher les données extraites des différentes tables, chacune sur une ligne, en créant un champs supplémentaire indiquant la source des données.

Id\_user      

"12345"	"A"	"11"	"3"	"4"	"1"
"12345"	"B"	"6336"	"Unit 1: Same Value Q"	"cs101 value same"	"question"    "\N"    "\N"

Source des données (A si de la table 1, B si de la table 2)

- Reducer :
  - Fera l'opération de jointure entre les deux sources, en testant la provenance avec le champs supplémentaire (A ou B)

---

Hadoop & Map Reduce

EN CONCLUSION...

---

# Hadoop...

## Avantages

---

- **La gestion des défaillances** : que ce soit au niveau du stockage ou traitement, les nœuds responsables de ces opérations durant le processus de Hadoop sont automatiquement gérés en cas de défaillance. Nous avons donc une forte tolérance aux pannes.
- **La sécurité et persistance des données** : Grâce au concept « Rack Awareness », il n'y a plus de soucis de perte de données.
- **La montée en charge**: garantie d'une montée en charge maximale.
- **La complexité réduite**: capacité d'analyse et de traitement des données à grande échelle.
- **Le coût réduit** : Hadoop est open source, et malgré leur massivité et complexité, les données sont traitées efficacement et à très faible coût

# Hadoop...

## Inconvénients

---

- Difficulté d'intégration avec d'autres systèmes informatiques: le transfert de données d'une structure Hadoop vers des bases de données traditionnelles est loin d'être trivial
- Administration complexe : Hadoop utilise son propre langage. L'entreprise doit donc développer une expertise spécifique Hadoop ou faire appel à des prestataires extérieurs
- Traitement de données différé et temps de latence important: Hadoop n'est pas fait pour l'analyse temps réel des données.
- Produit en développement continu: manque de maturité

# Sources

---

- Cours
  - Big Data Analytics – Lesson 1: What is Big Data, IBM, Big Data University
  - Intro to Hadoop and MapReduce, Coursera, Udacity
- Présentations
  - Introduction to YARN and MapReduce2, Cloudera
  - HDFS, Formation Big Data Specialist - IBM