Institut National des Sciences Appliquées et de Technologie

Big Data

Chp2 – Hadoop et MapReduce

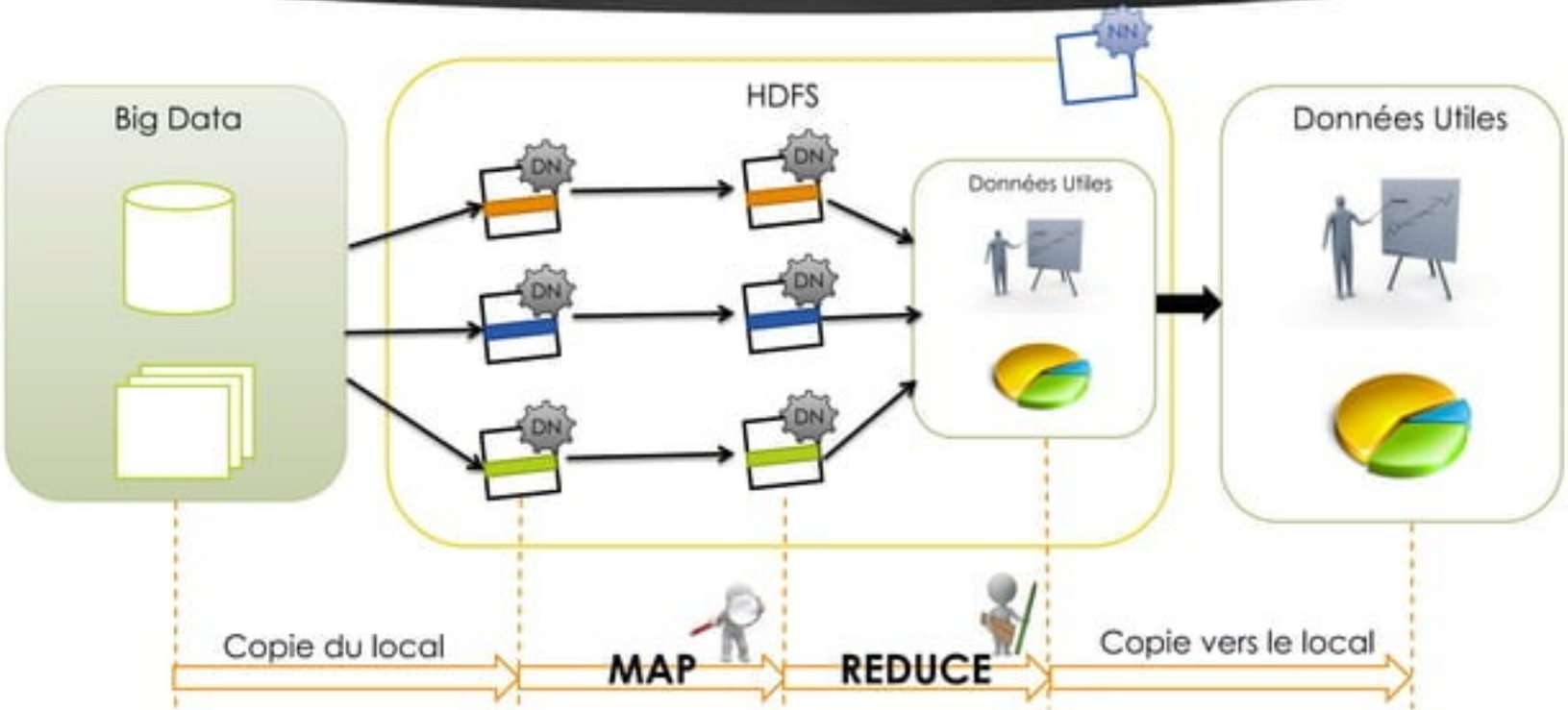
Dr. Lilia SFAXI

GL4-OPTION MANAGEMENT DES SYSTÈMES D'INFORMATION - 2013-2014

Hadoop

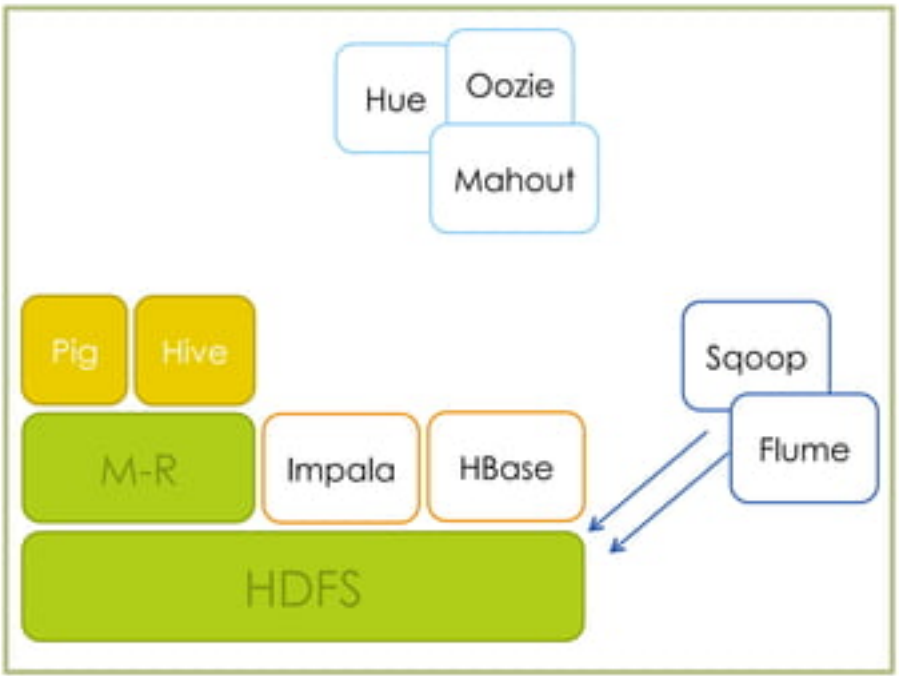
- Le projet Hadoop consiste en deux grandes parties:
  - Stockage des données : **HDFS** (***H**adoop **D**istributed **F**ile **S**ystem*)
  - Traitement des données : **MapReduce**
- Principe :
  - Diviser les données
  - Les sauvegarder sur une collection de machines, appelées *cluster*
  - Traiter 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



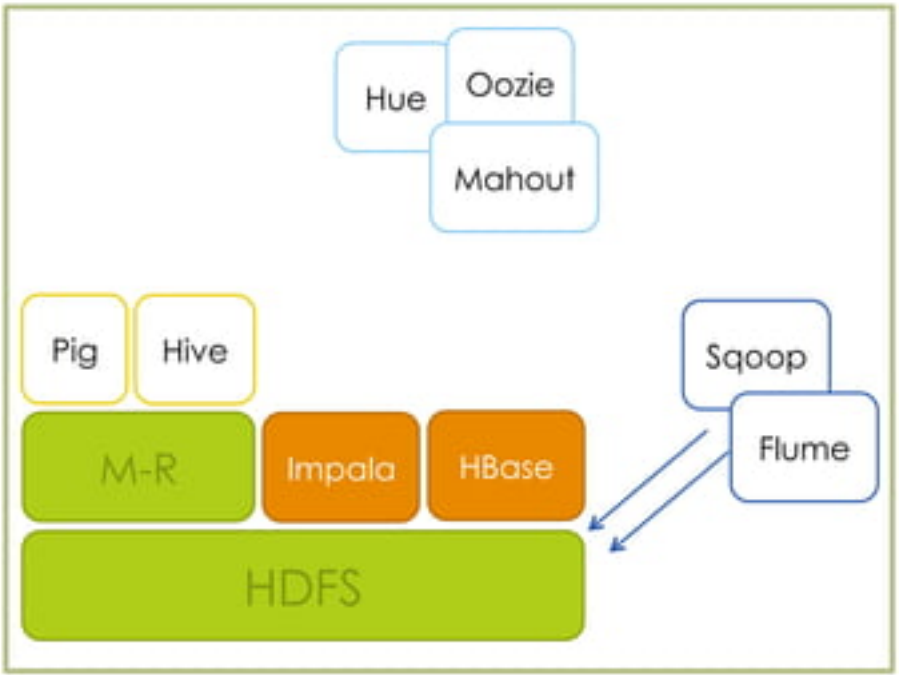
Ecosystème de Hadoop

- MapReduce utilise des langages de programmation pour traiter les données
  - Java, Ruby, Python...
- Plusieurs outils facilitant le travail
- **Au dessus du MapReduce** : langage plus simple traduit plus tard en Mappers et Reducers
  - **PIG** : script simple
  - **Hive** : requêtes SQL



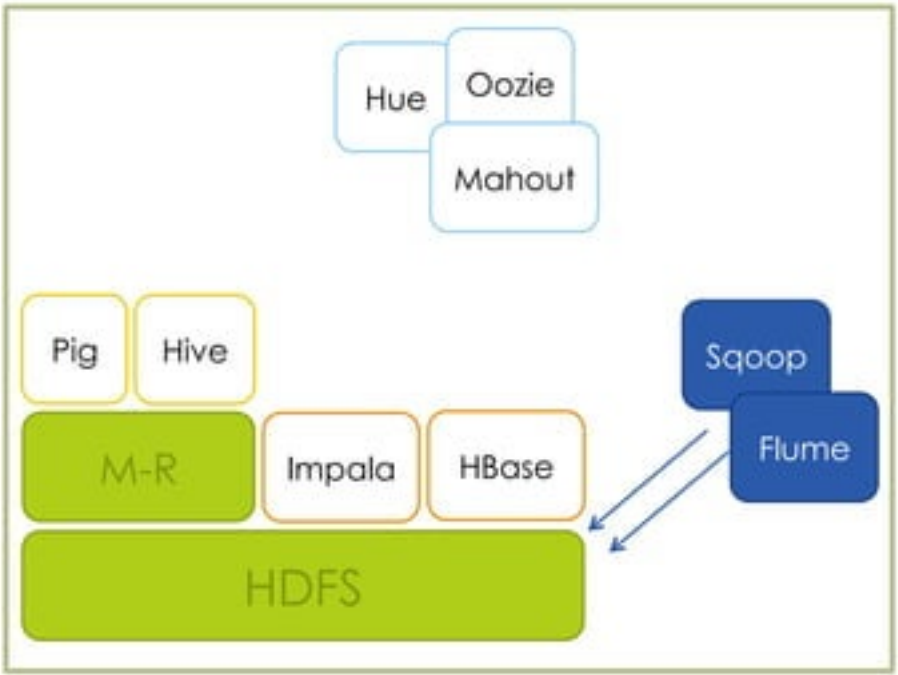
Ecosystème de Hadoop

- Les jobs MapReduce peuvent prendre beaucoup de temps pour s'exécuter sur de larges quantités de données
  - Autres projets pour simplifier
- **Impala** :
  - Extraction des données directement à partir de HDFS avec SQL
  - Optimisé pour les requêtes à faible latence
  - Requêtes plus rapides que Hive
- **HBase** :
  - Base de données temps réél



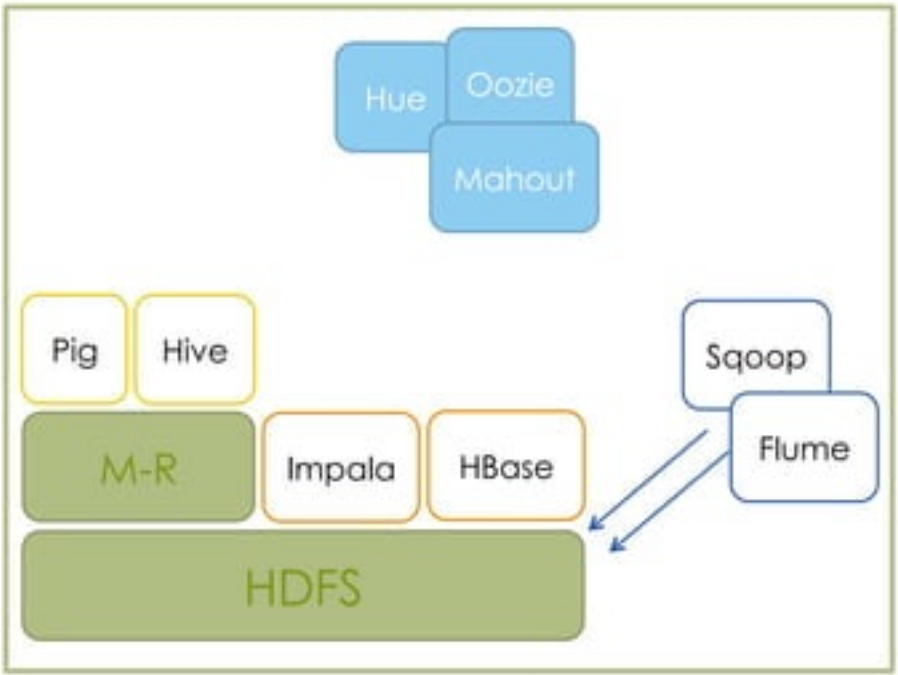
Ecosystème de Hadoop

- Connexion du HDFS à partir d'outils externes
- **Sqoop :**
  - Prend les données à partir d'une base de données traditionnelle, et les met dans HDFS, comme étant des fichiers délimités, pour être traitées avec d'autres données dans le cluster
- **Flume :**
  - Système distribué permettant de collecter, regrouper et déplacer efficacement un ensemble de données (des logs) à partir de plusieurs sources vers le HDFS



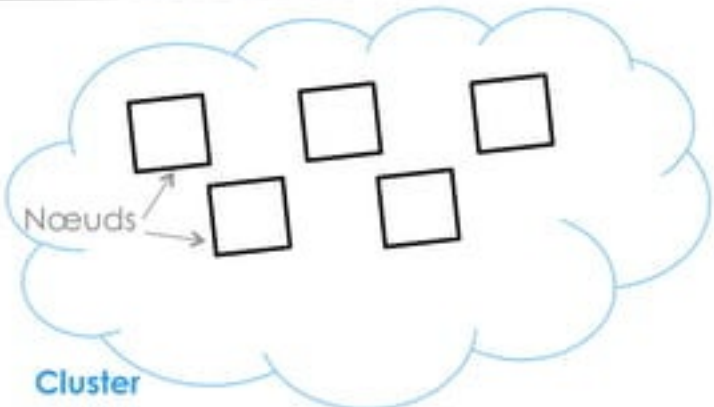
Ecosystème de Hadoop

- **Hue :**
  - Front-end graphique pour le cluster
  - Fournit
    - Un navigateur pour HDFS et HBase
    - Des éditeurs pour Hive, Pig, Impala et Sqoop
- **Oozie :**
  - Outil de gestion de Workflow
  - Permet de gérer les jobs Hadoop
- **Mahout :**
  - Bibliothèque d'apprentissage automatique
  - Permet de :
    - Déterminer des éléments qu'un utilisateur pourra apprécier selon son comportement
    - Grouper des documents
    - Affecter automatiquement des catégories aux documents



HDFS : Hadoop Distributed File System

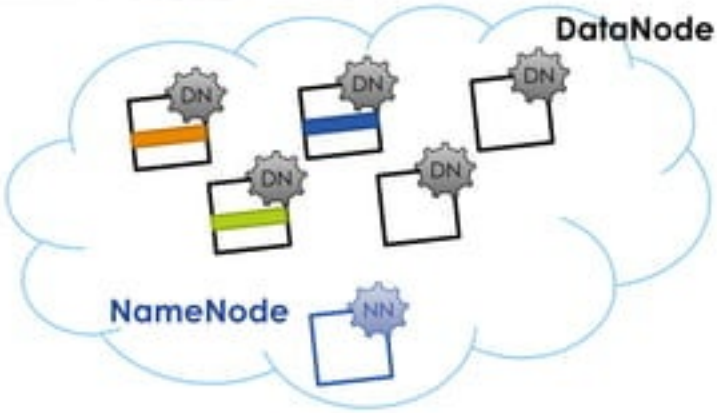
- HDFS est un système de fichiers distribué, extensible et portable
- Ecrit en Java
- Permet de stocker de très gros volumes de données sur un grand nombre de machines (nœuds) équipées de disques durs banalisés ➔ Cluster
- Quand un fichier *mydata.txt* est enregistré dans HDFS, il est décomposé en grands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



mydata.txt (150 Mo)

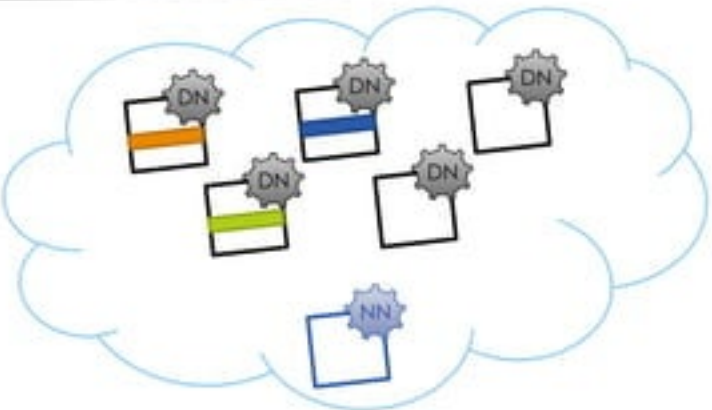
HDFS : Hadoop Distributed File System

- 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



HDFS : Hadoop Distributed File System

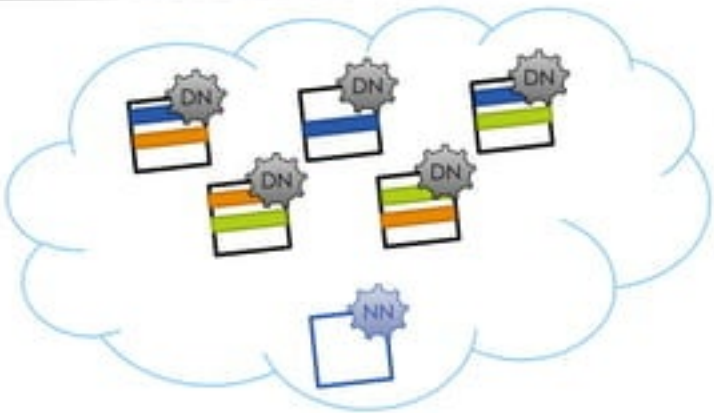
- Quels sont les problèmes possibles?
  - ✔ Panne de réseau ?
  - ✔ Panne de disque sur les DataNodes ?
  - ❑ Pas tous les DN sont utilisés ?
  - ❑ Les tailles des blocks sont différentes ?
  - ✔ Panne de disque sur les NameNodes ?



mydata.txt (150 Mo)

HDFS : Hadoop Distributed File System

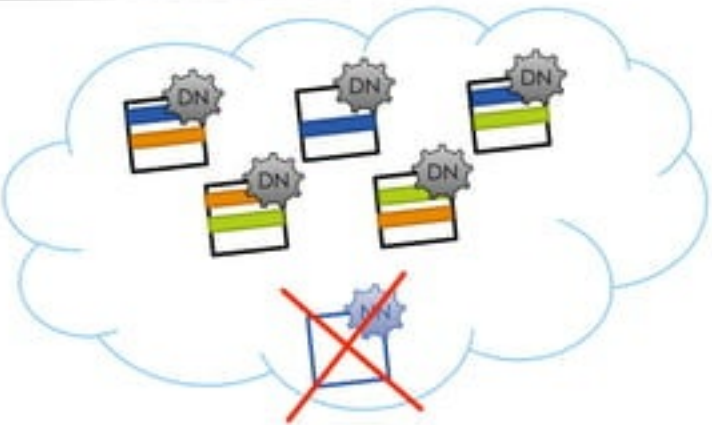
- Si l'un des nœuds a un problème, les données seront perdues
  - Hadoop réplique chaque bloc 3 fois
  - Il choisit 3 nœuds au hasard, et place une copie du bloc dans chacun d'eux
  - Si le nœud est en panne, le NN le détecte, et s'occupe de répliquer encore les blocs qui y étaient hébergés pour avoir toujours 3 copies stockées
  - Concept de **Rack Awareness** (rack = baie de stockage)
- Si le NameNode a un problème ?



mydata.txt (150 Mo)

HDFS : Hadoop Distributed File System

- Si le NameNode a un problème :
  - Données inaccessibles?
  - ✔ Données perdues à jamais
  - ✔ Pas de problème
- 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

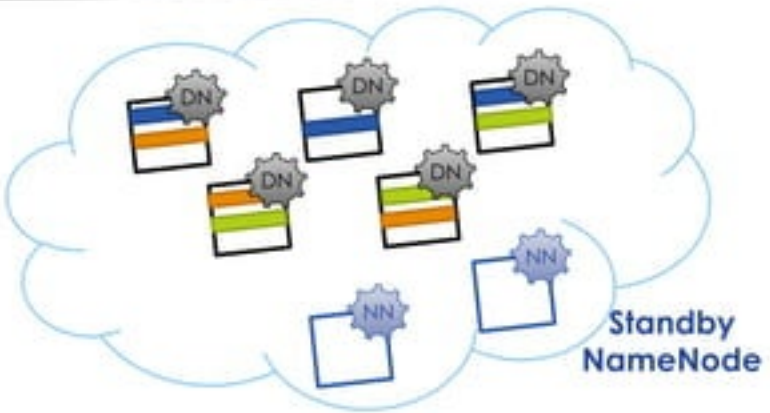


mydata.txt (150 Mo)

HDFS : Hadoop Distributed File System

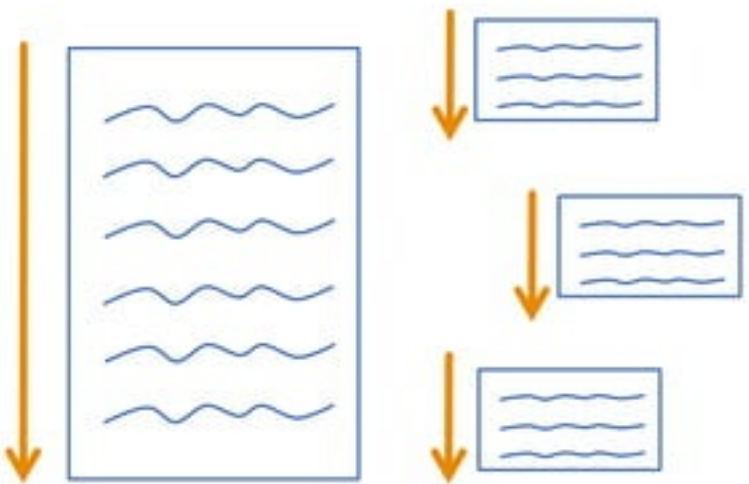
- Pour éviter cela, le NameNode sera dupliqué, non seulement sur son propre disque, mais également quelque part sur le système de fichiers du réseau

- Définition d'un autre NN (*standby namenode*) pour reprendre le travail si le NameNode actif est défaillant



Map-Reduce

- 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:
  - Jour Ville produit Prix

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

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

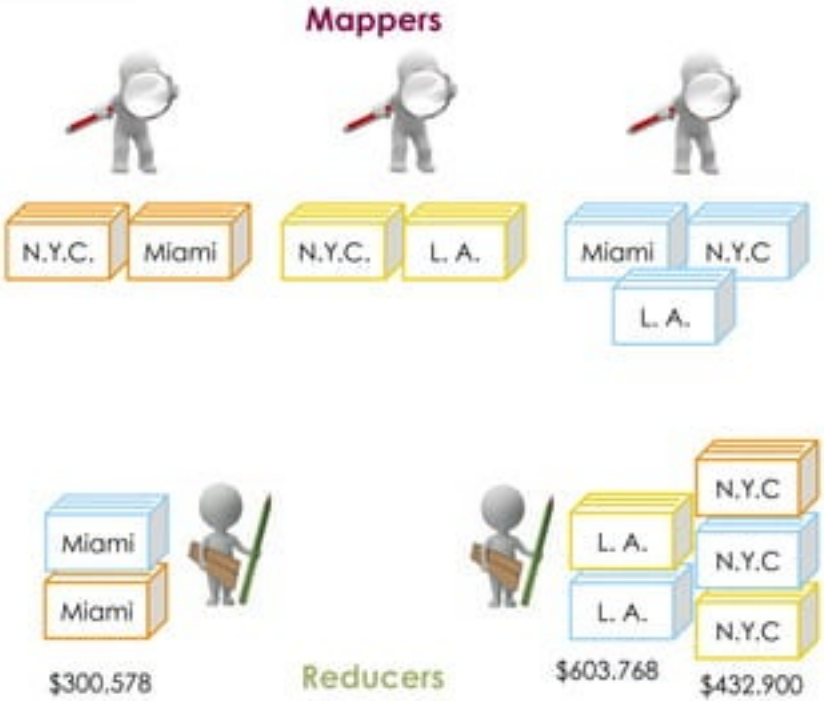
Map-Reduce : Exemple

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

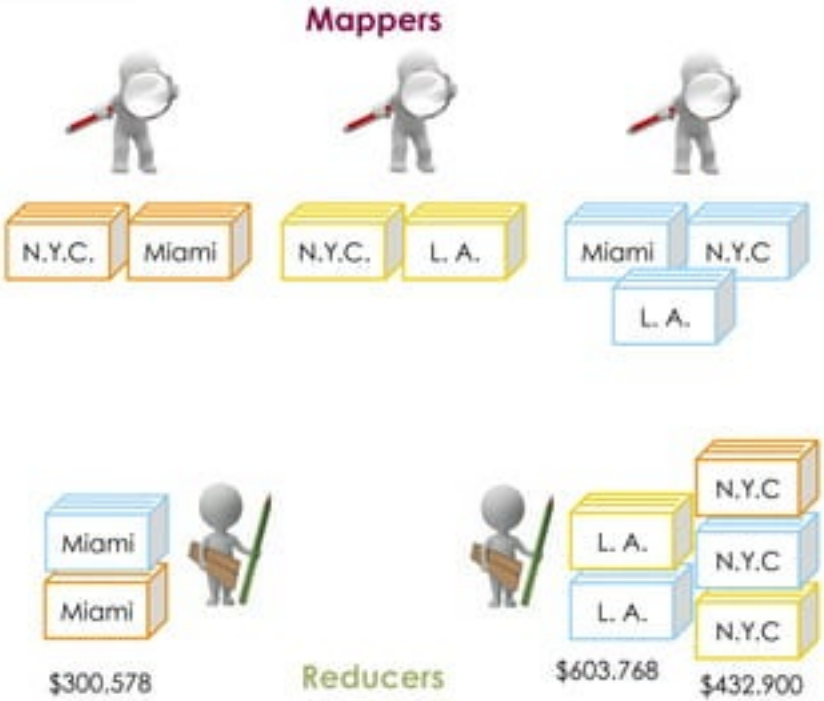


Map-Reduce : Exemple

- Le Reducer reçoit des données comme suit :

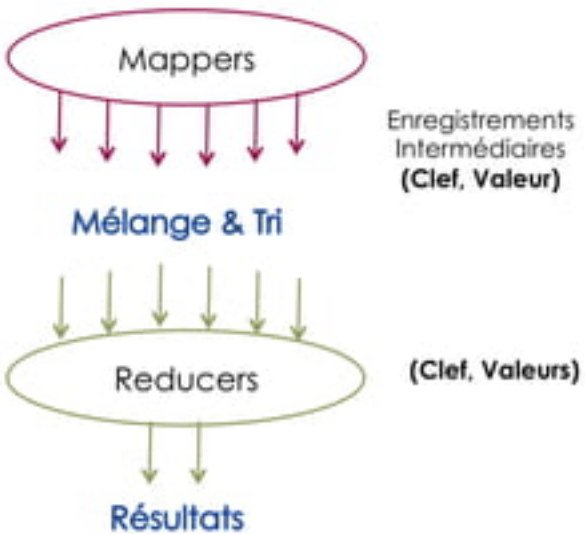
Miami	12.34
Miami	99.07
Miami	3.14
NYC	99.77
NYC	88.99

- 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



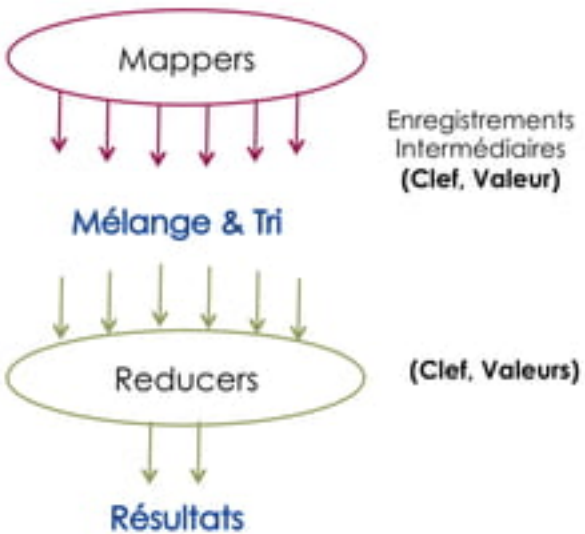
Map-Reduce : Fonctionnement

- Les Mappers sont de 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



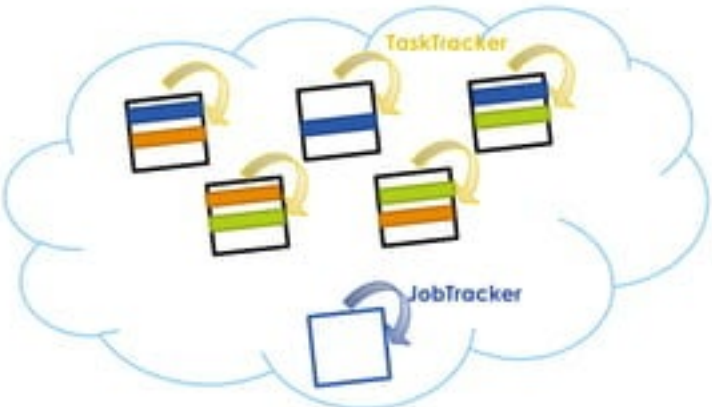
Map-Reducer : Résultats

- 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.



Démons de MapReduce

- **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 Mape-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)

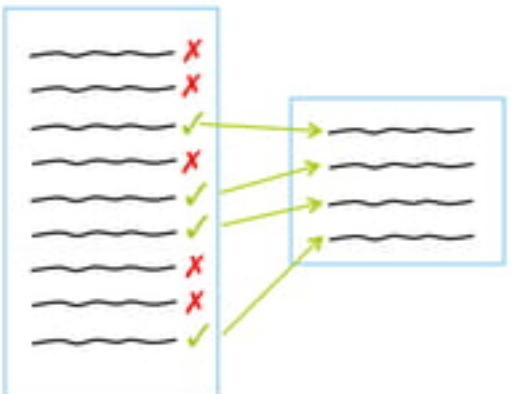


Hadoop Design Patterns

- 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 enregistrement
    - ✓ Trouver les min et les max
    - ✓ Statistiques
    - ✓ Indexes
  - Patrons Structurels
    - ✓ Combinaison de données relationnelles

Patrons de Filtrage

- 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
Exemples

- 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
Exemples

- 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

- 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

- 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.

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

Atouts de Hadoop

- 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 Awarness* », 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

Inconvénients de Hadoop

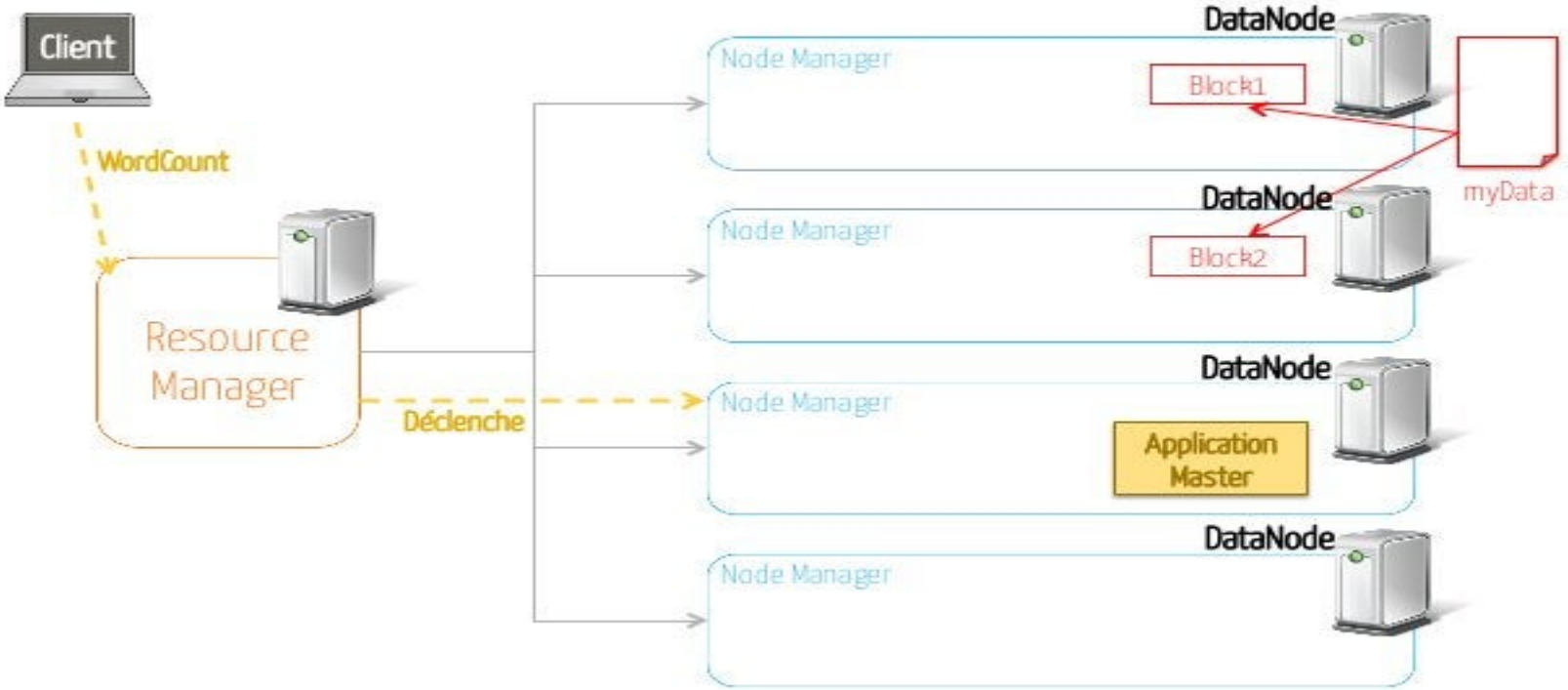
- **Difficulté d'intégration avec d'autres systèmes informatiques**: le transfert de données d'une structure Hadoop vers des bases de données traditionnelles est loin d'être trivial
- **Administration complexe** : Hadoop utilise son propre langage. L'entreprise doit donc développer une expertise spécifique 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 réé 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

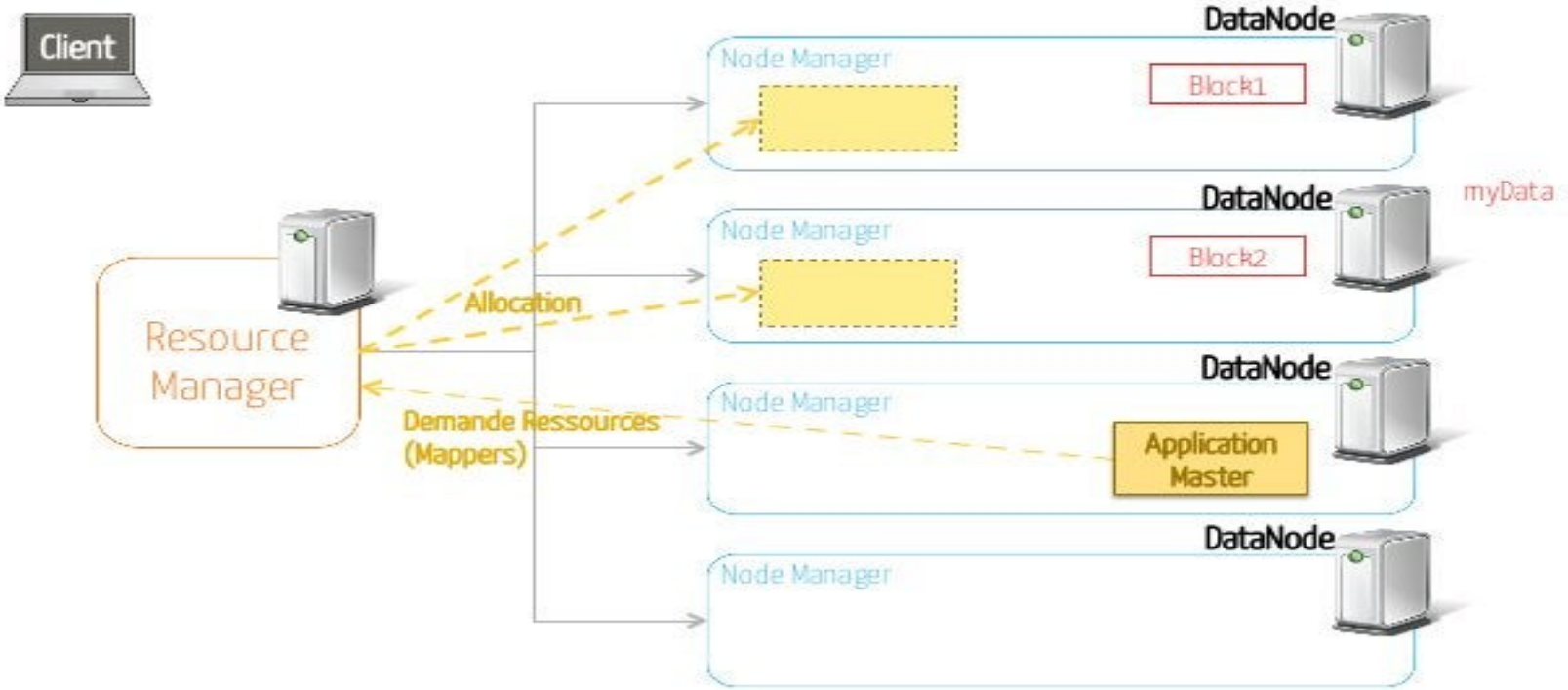
MapReduce V2 (MRv2)

Exécution d'un Job Map-Reduce



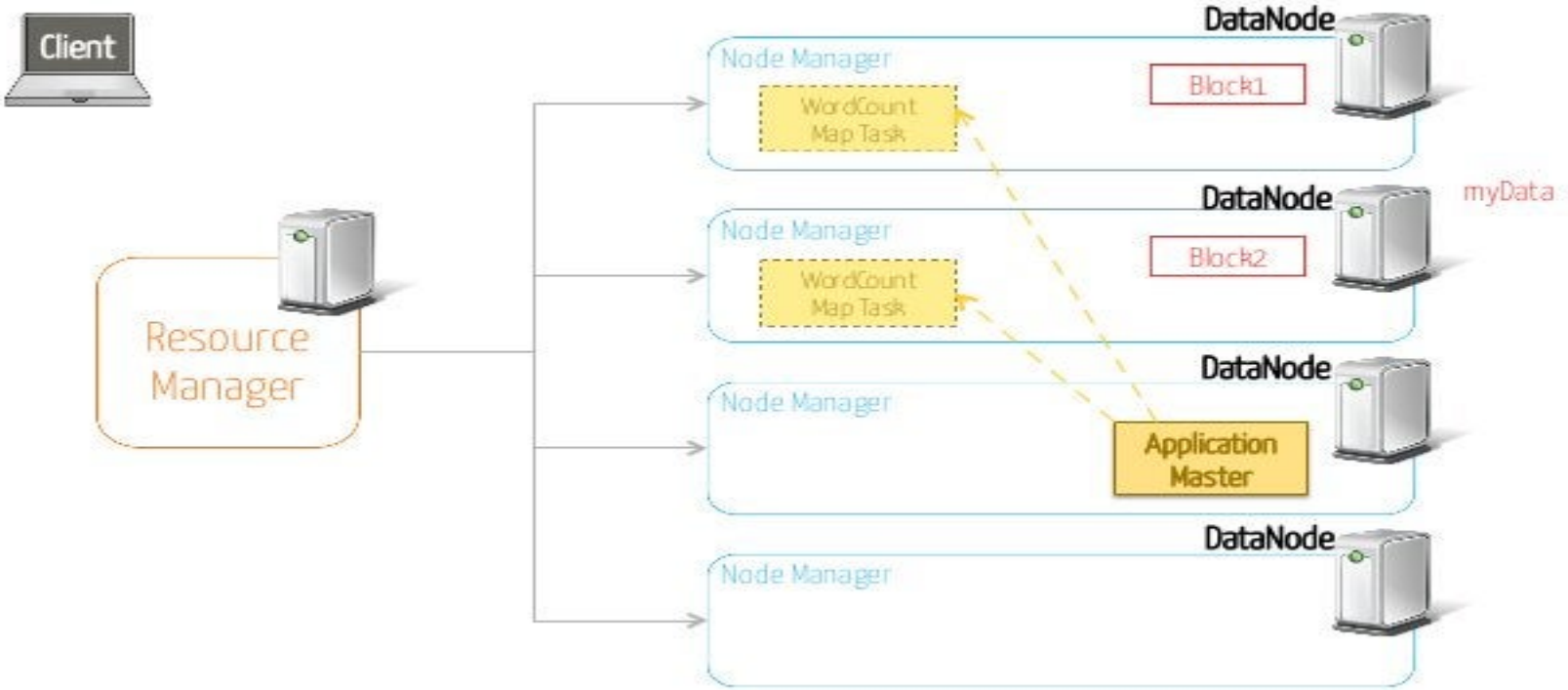
MapReduce V2 (MRv2)

Exécution d'un Job Map-Reduce



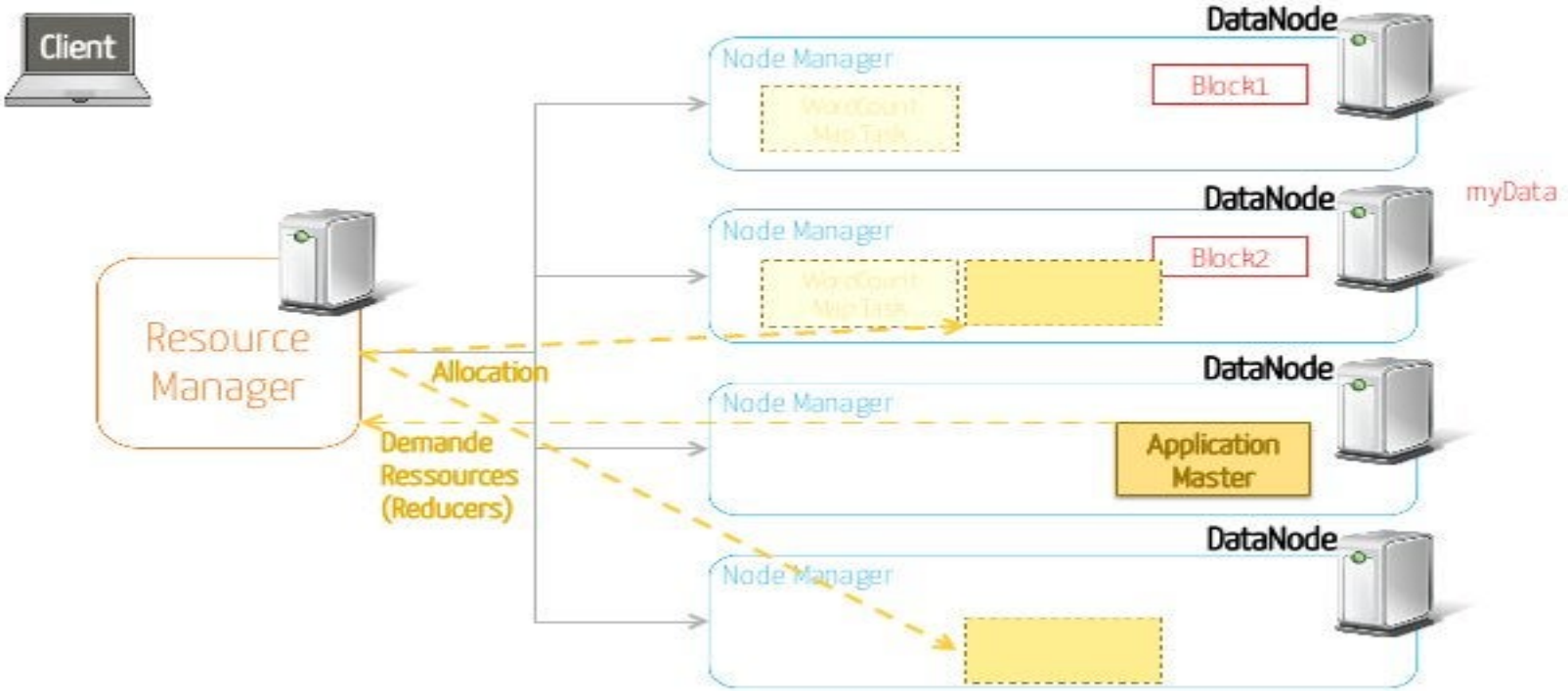
MapReduce V2 (MRv2)

Exécution d'un Job Map-Reduce



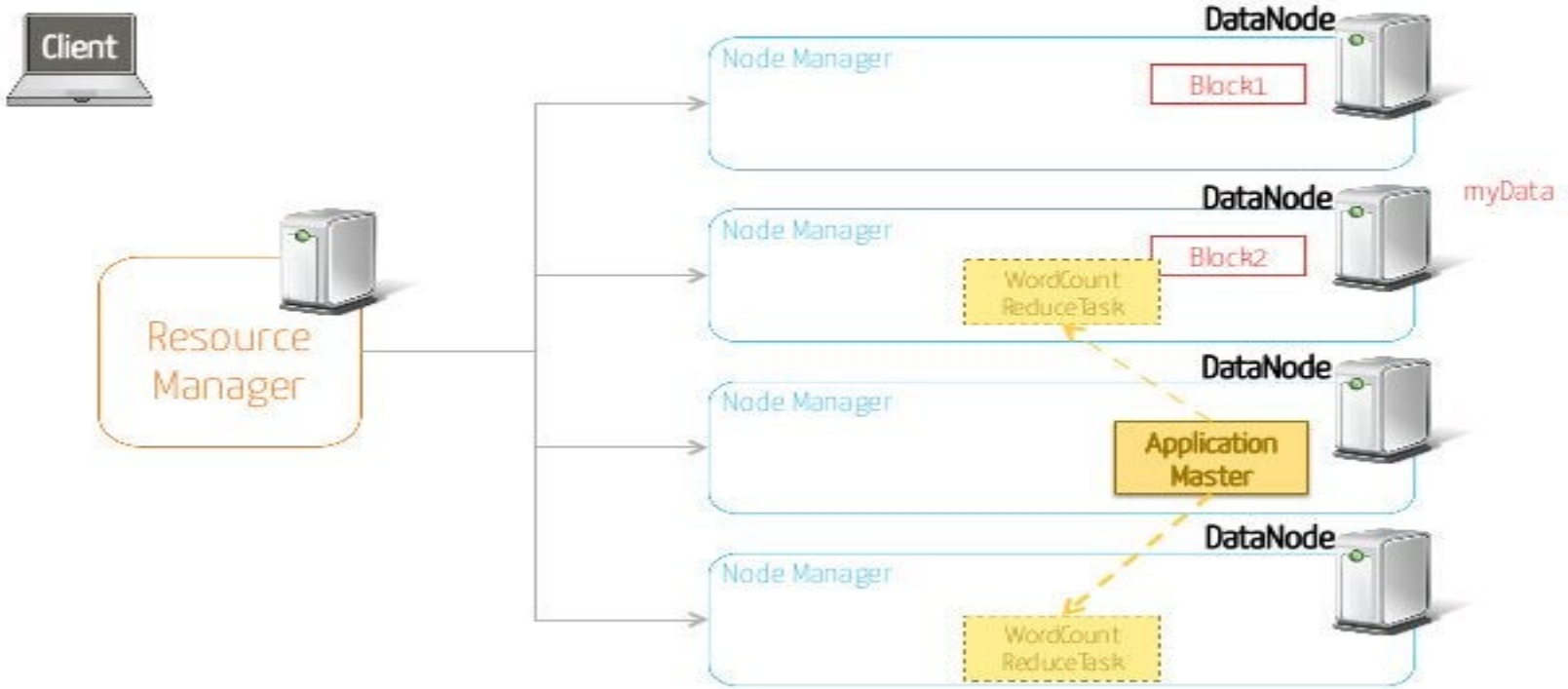
MapReduce V2 (MRv2)

Exécution d'un Job Map-Reduce



MapReduce V2 (MRv2)

Exécution d'un Job Map-Reduce



MAP-REDUCE DESIGN PATTERNS

Map-Reduce Design Patterns

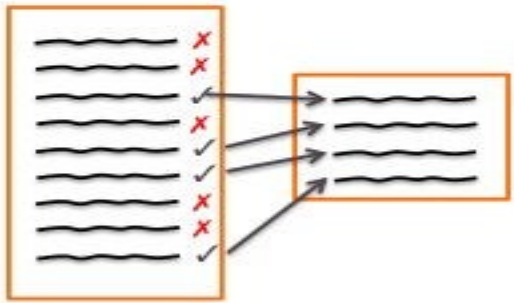
Présentation

- Les designs patterns (Patrons de Conception) représentent les types de traitements Map-Reduce 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

Exemple

- 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

Exemple

- 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.

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

EN CONCLUSION...

Hadoop...
Avantages

- **La gestion des défaillances** : que ce soit au niveau du stockage ou traitement, les nœuds responsables de ces opérations 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 Awarness », 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