Hadoop Map Reduce

Dr. Houda EL BOUHISSI

M1 GL

Hadoop -Rappel

Le projet Hadoop consiste en deux grandes parties:

- Stockage des données : HDFS (Hadoop Distributed File System)
- Traitement des données : MapReduce / Yarn

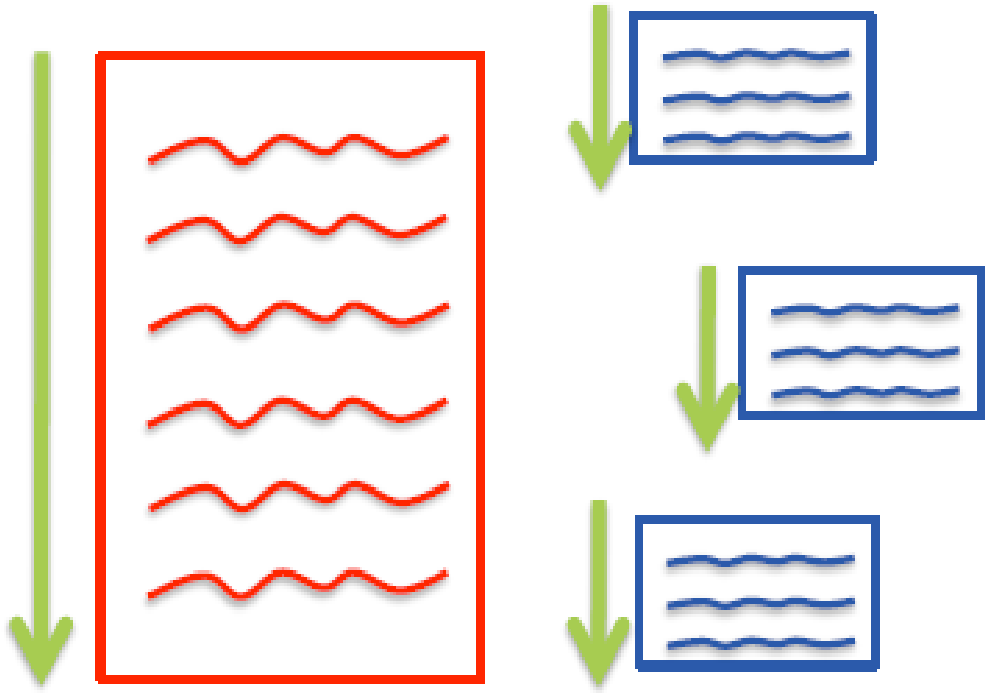
Principe :

- 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

Map-Reduce - Définition

Patron d'architecture de développement permettant de traiter des données volumineuses de manière parallèle et distribuée.

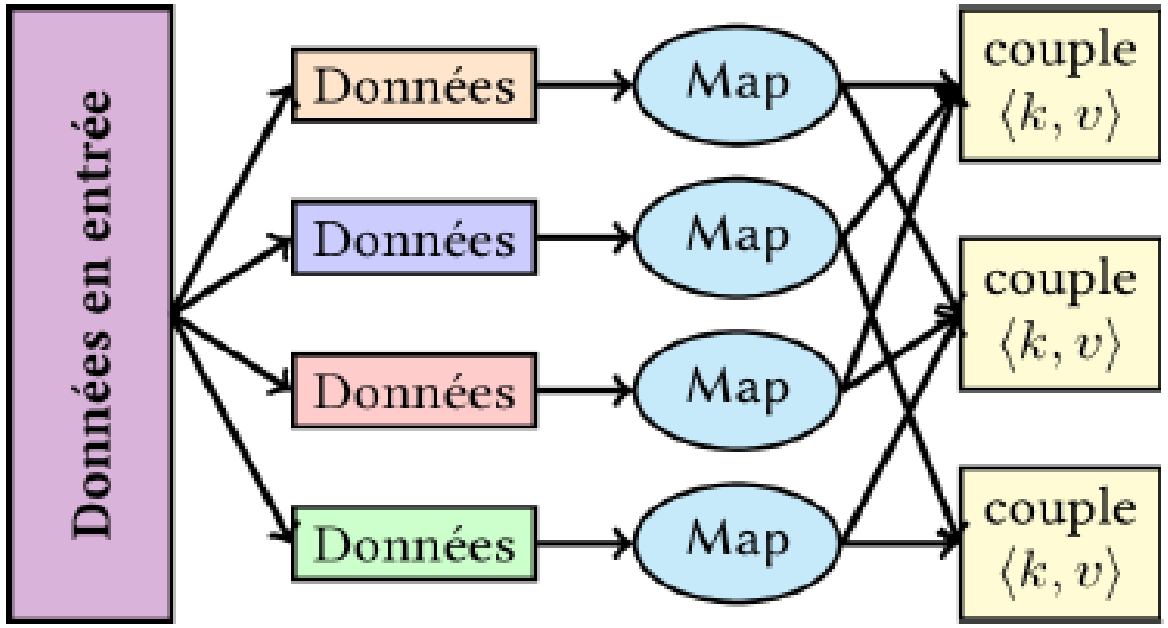
- A la base, le langage Java est utilisé, mais grâce à une caractéristique de Hadoop appelée Hadoop Streaming, il est possible d'utiliser d'autres langages comme Python ou Ruby.
- Au lieu de parcourir le fichier séquentiellement (bcp de temps), il est divisé en morceaux qui sont parcourus en parallèle.



Opération Map

Transforme les données d'entrée en série de couples clé/valeur. Les couples clés/valeurs doivent avoir un sens par rapport au problème à résoudre.

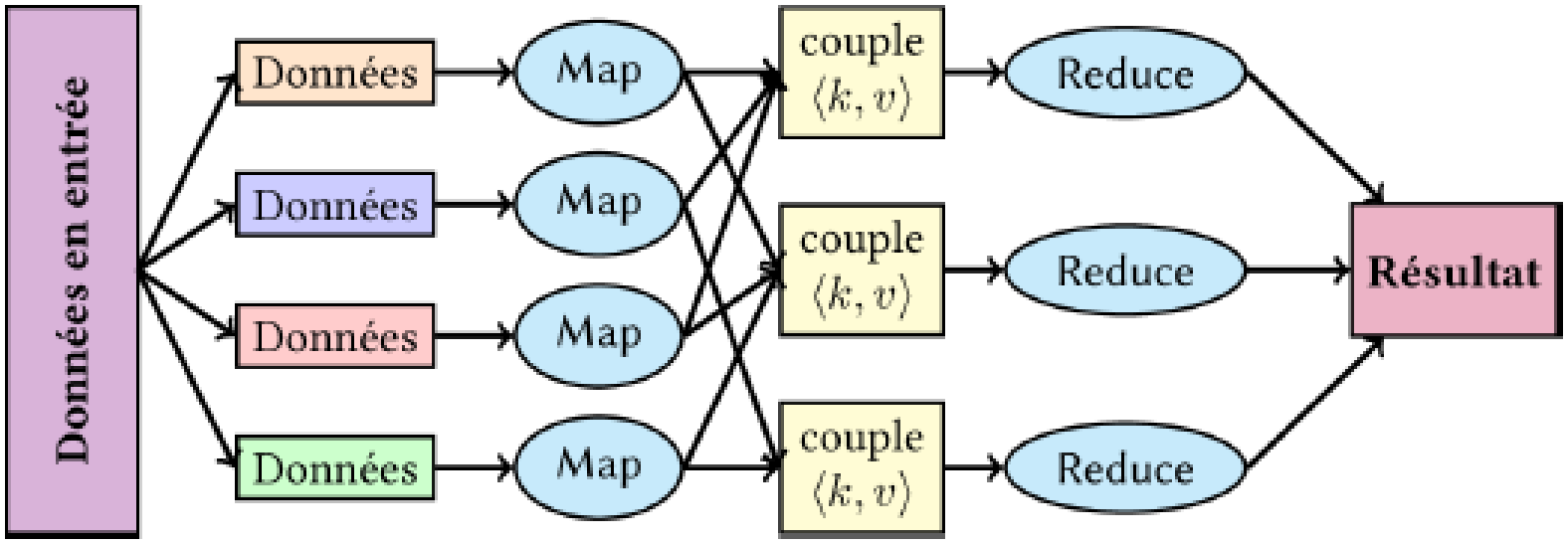
Les données doivent être découpées en plusieurs fragments et l'opération **Map** est exécutée sur chaque machine du cluster sur un fragment distinct.



Opération Reduce

Applique un traitement à toutes les valeurs de chacune des clés distinctes produites par l'opération Map.

- Pour chaque machine du cluster est attribué une des clés uniques produites par l'opération Map, en lui donnant la liste des valeurs associées à cette clé.
- Chacune des machines effectue l'opération **Reduce** pour cette clé.



Map-Reduce - Méthodologie

- Choisir une manière de découper les données d'entrée pour pouvoir paralléliser l'opération Map.
- Définir quelle clé utiliser pour le problème à résoudre.
- Écrire le programme pour l'opération Map.
- Écrire le programme pour l'opération Reduce.

Map-Reduce - Étapes de traitements

Un traitement MapReduce est caractérisé par 4 étapes distinctes :

Split : Découper les données d'entrée en plusieurs fragments.

Map : Mapper chacun de ces fragments pour obtenir des couples (clé/valeur).

Shuffle : Grouper ces couples par clé.

Reduce : Réduire les groupes indexés par clé en une forme finale, avec une valeur pour chacune des clés distinctes.

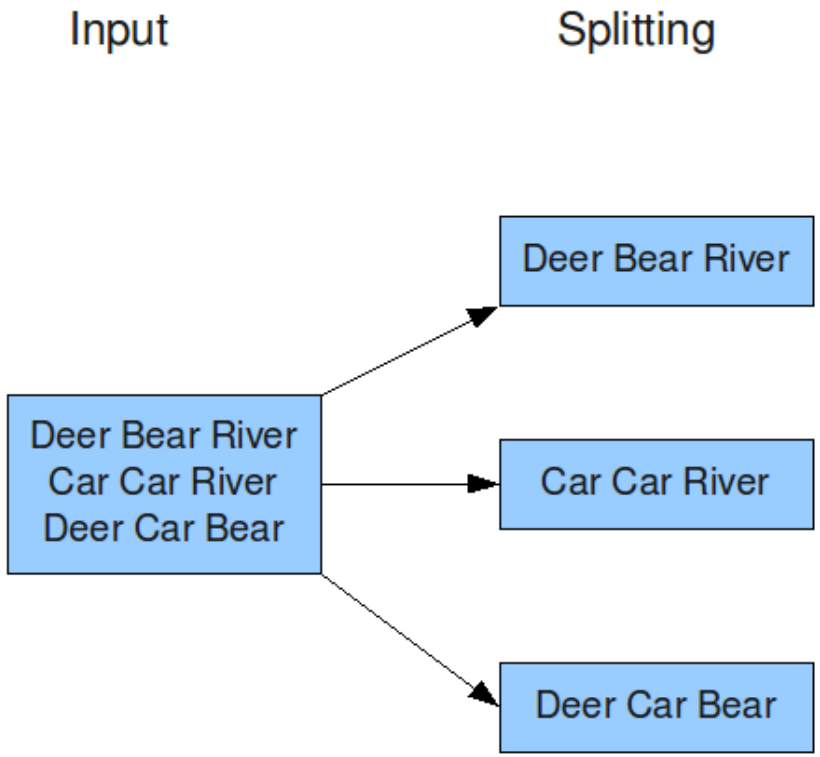
Chacune des tâches (à l'exception de la première) seront effectuées de manière distribuée.
Le calcul distribué, groupement par clé distincte... se font par Hadoop de manière transparente.

Exemple de WordCount

Compter le nombre d'occurrence des mots dans un document

Deer Bear River
Car Car River
Deer Car Bear

Étape 1 : Comment découper les données ?
Chacune des machines doit travailler sur une partie du texte
Exemple de découpage : ligne par ligne.

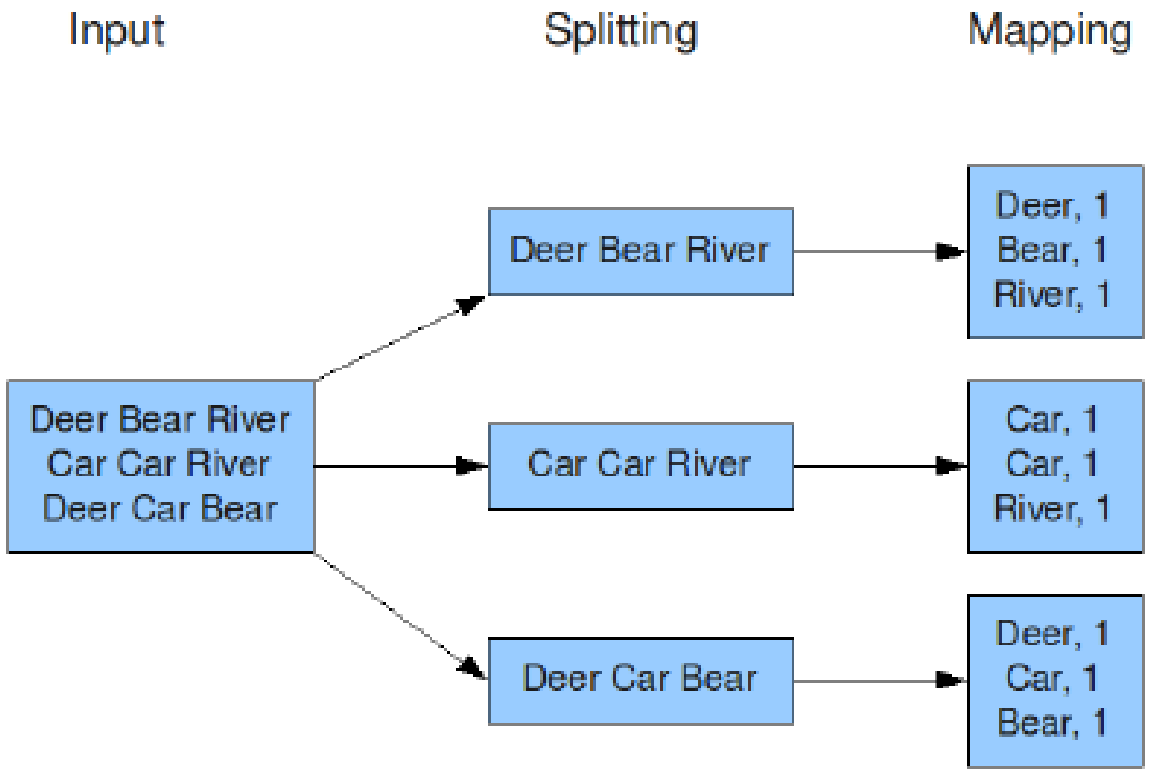


Exemple de WordCount

Étape 2 : Déterminer la clé et définir l'opération Map

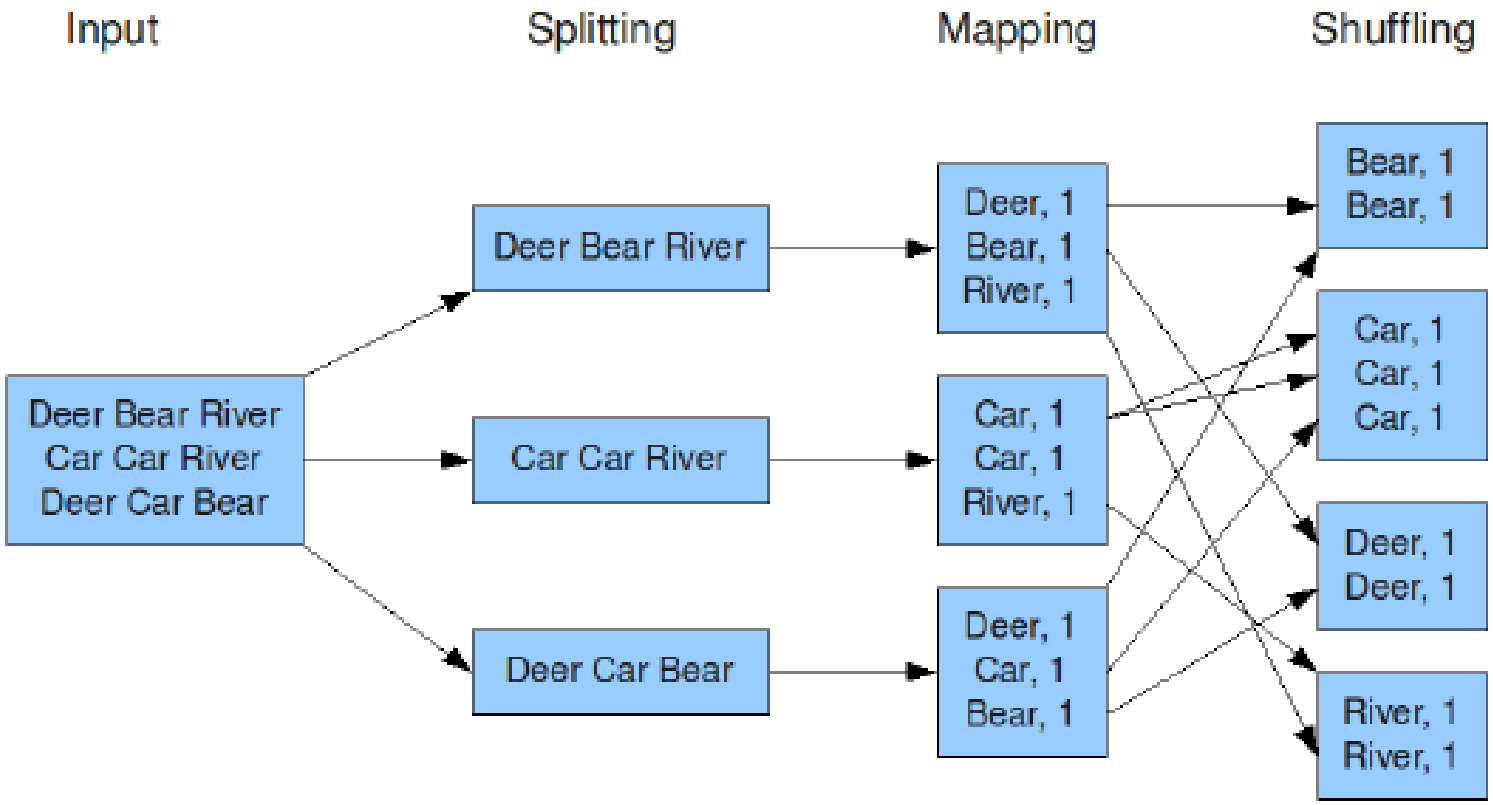
Clé : le mot lui-même.

Opération Map : Générer le couple clé/valeur : (mot, 1)



Exemple de WordCount

Étape 3 : Grouper tous les couples par clé commune (**Shuffle**)
Cette opération est effectuée automatiquement et de manière distribuée par Hadoop.

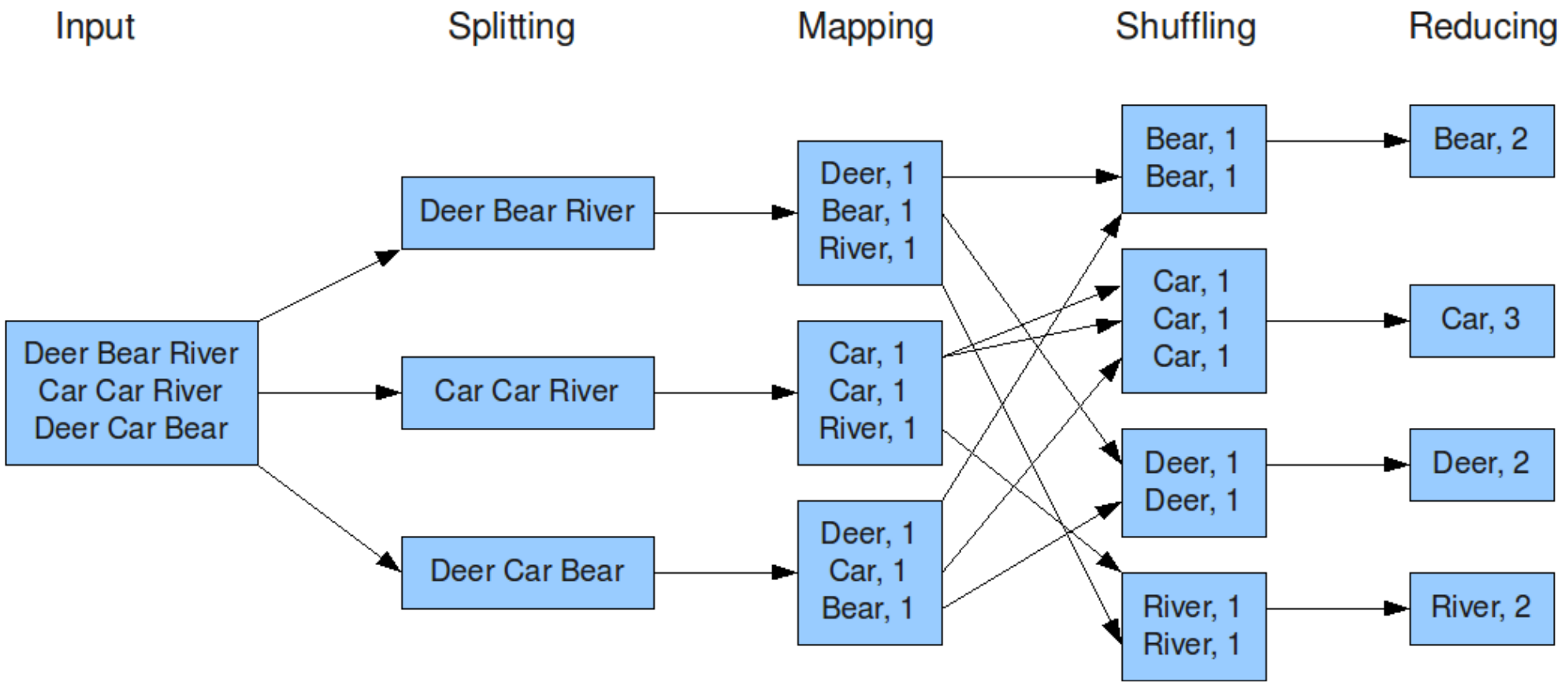


Exemple de WordCount

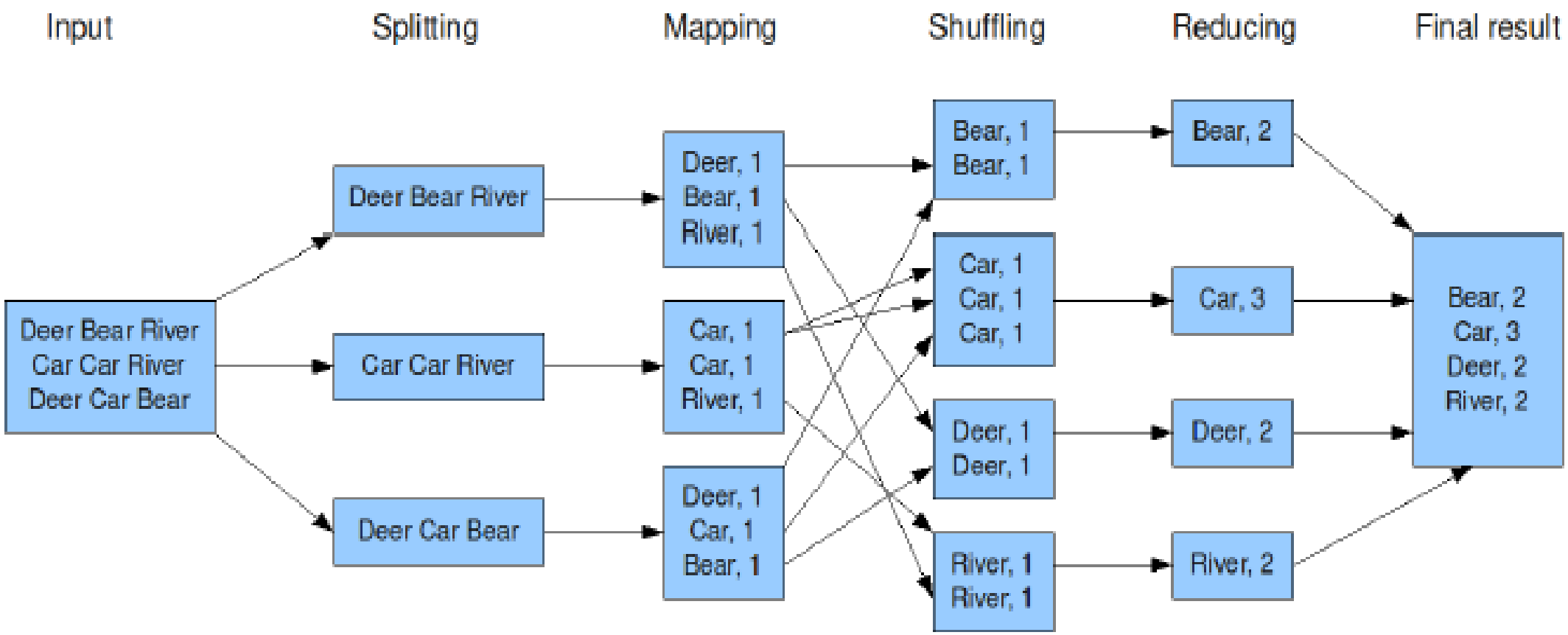
Étape 4 : Définir l'opération Reduce

Elle sera appliquée sur chacun des groupes par clé distincte.

Additionner toutes les valeurs liées à la clé spécifiée.



Exemple de WordCount – Schéma global



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

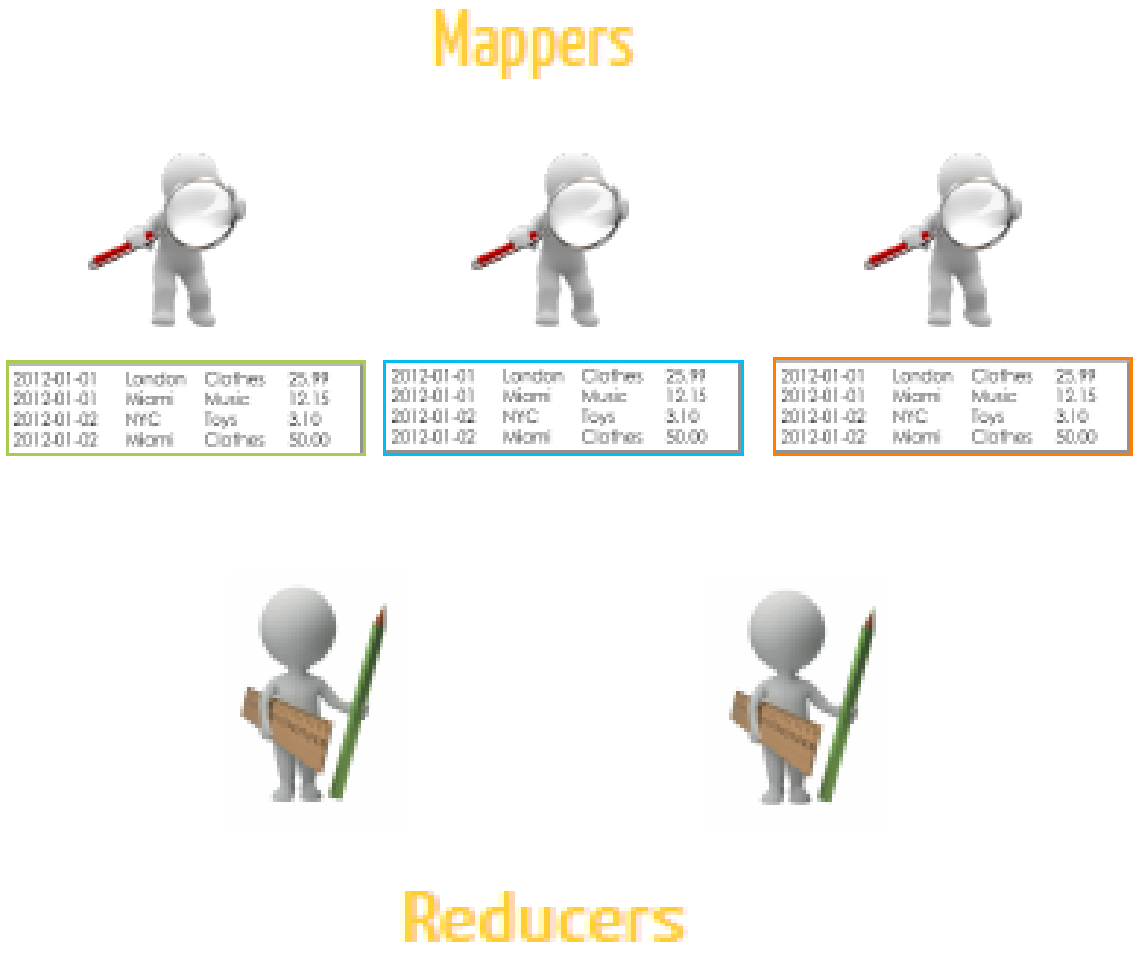
Map-Reduce : Moyen plus efficace et rapide de traiter ces données

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



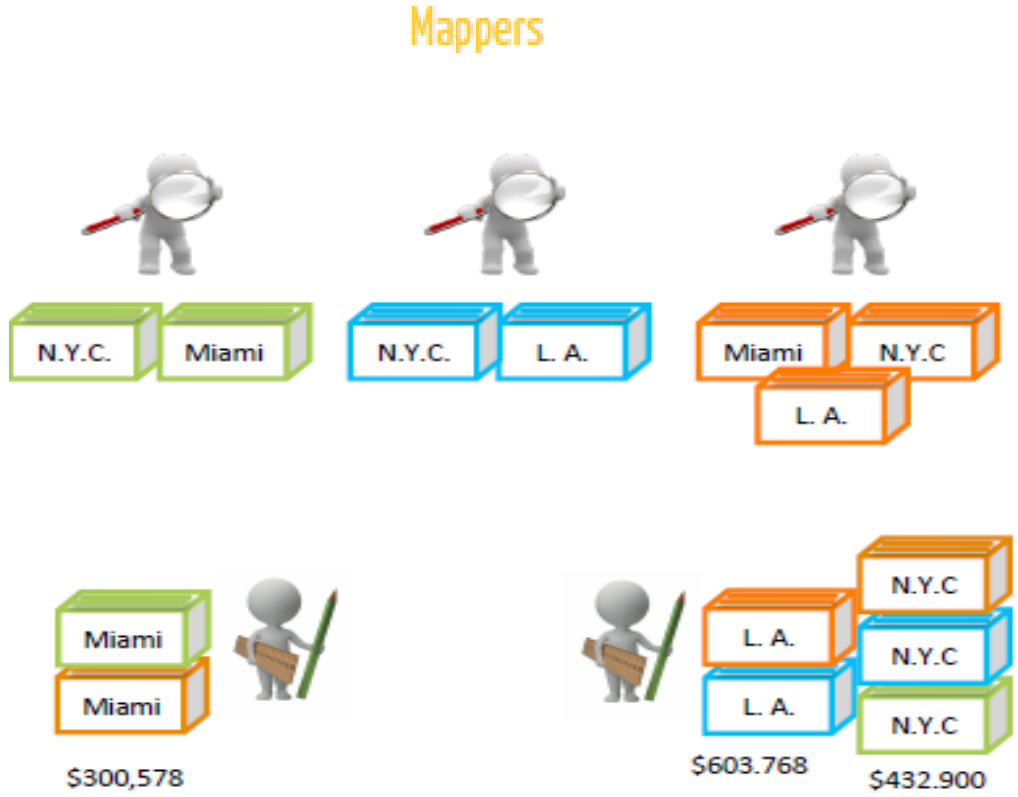
Map-Reduce

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



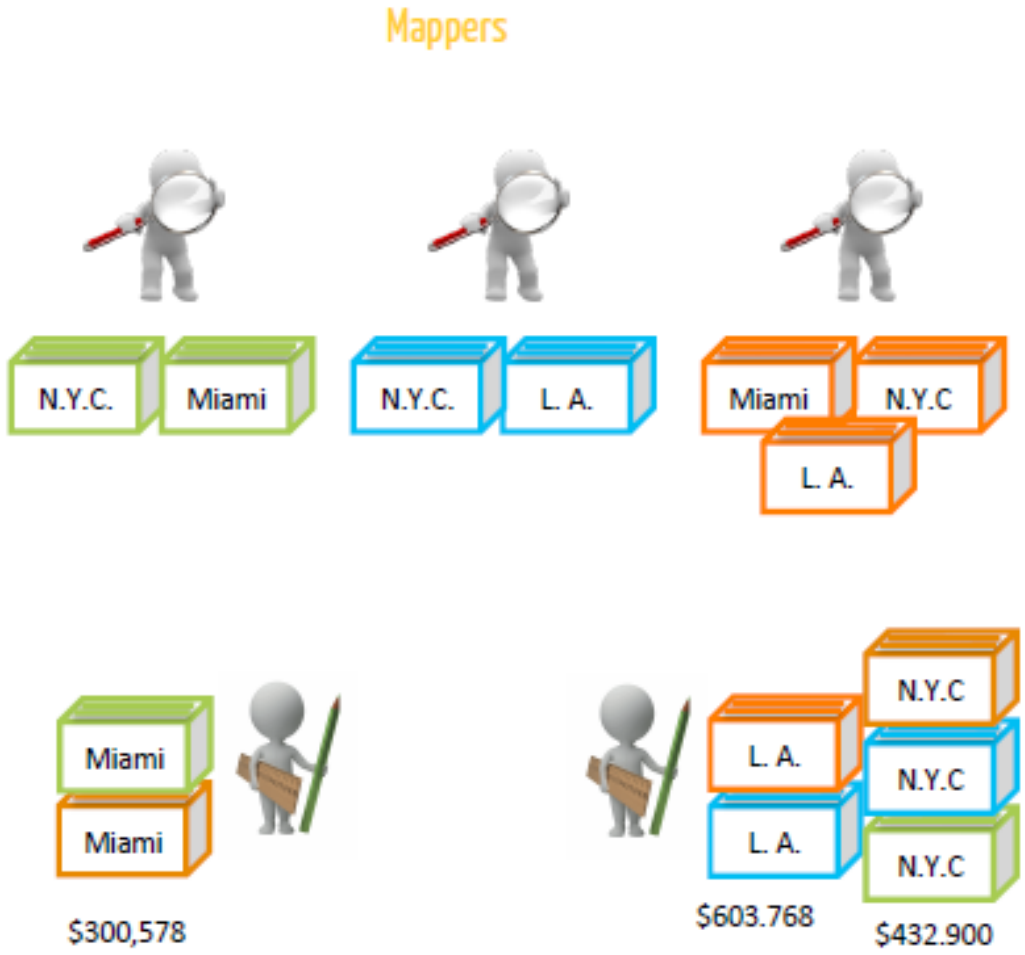
Map-Reduce

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?

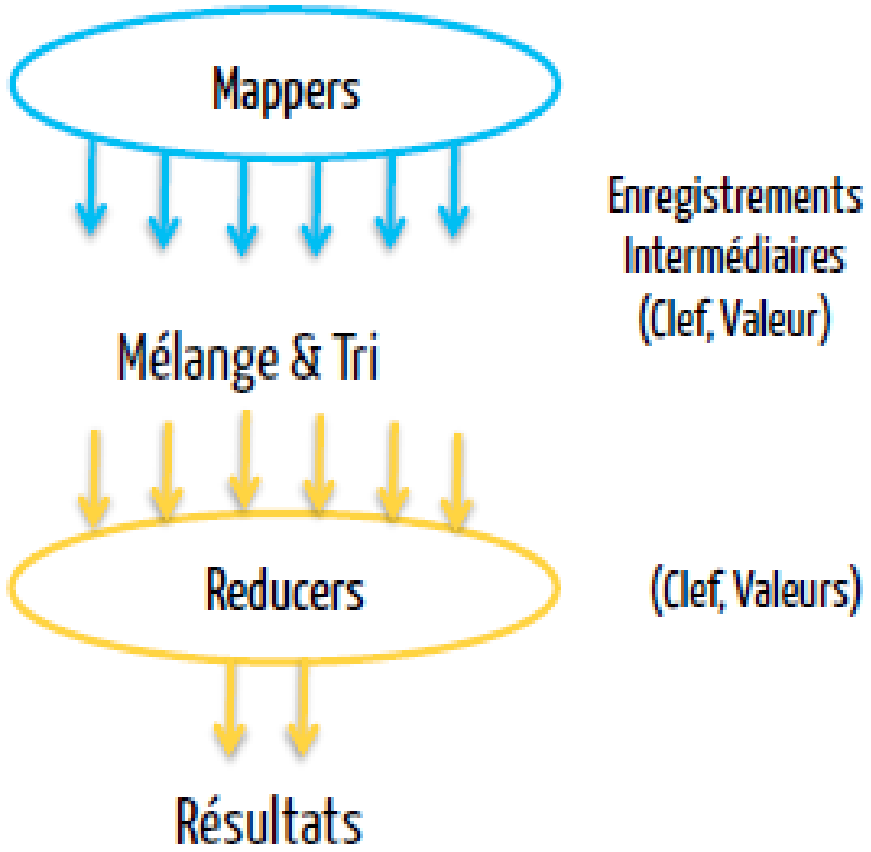
- Coût précédent
- Coût en cours
- Ventes totales par magasin
- Magasin précédent
- Magasin en cours



Map-Reduce

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.

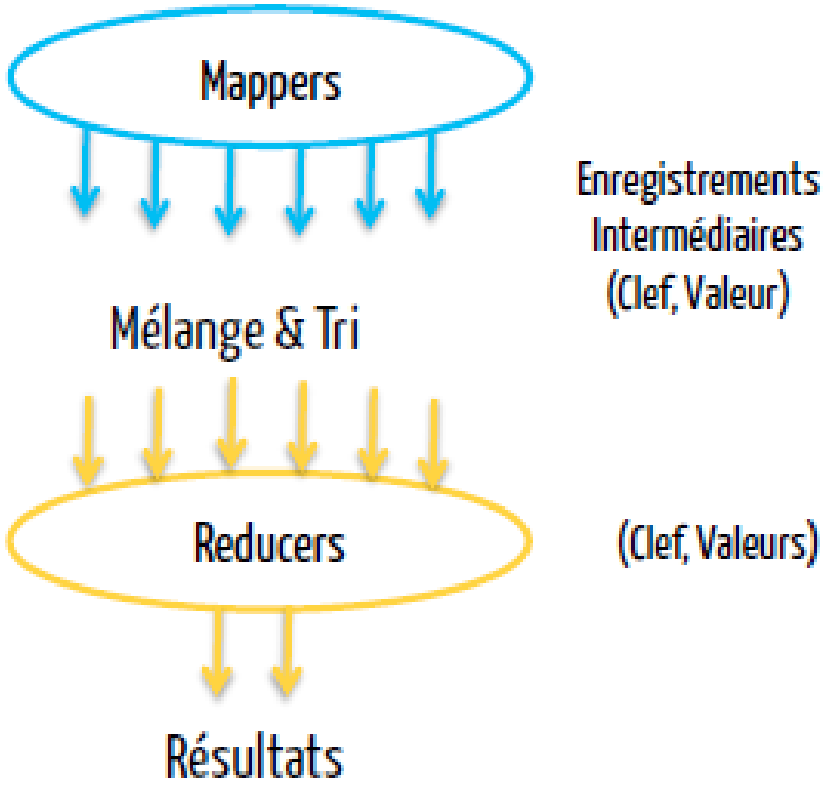


Map-Reduce

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

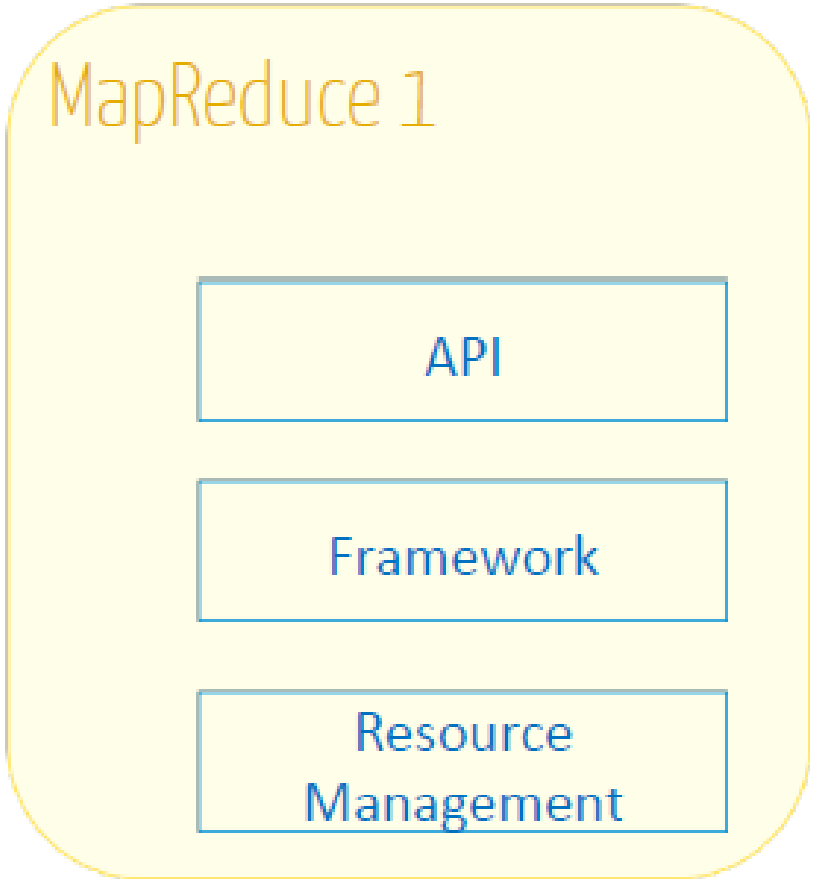
- Si on a plusieurs Reducers, on ne peut pas savoir lesquels traitent quelles clefs: le partitionnement est aléatoire.



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 nœuds 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 noeuds 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 sur plusieurs tâches appelées mappers et reducers.

- Chaque tâche est exécutée sur un nœud du cluster
- Chaque nœud 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 viceversa

- Le Job Tracker est fortement intégré à Map Reduce

§ Problème d'interoperabilité: impossible d'exécuter des applications non-MapReduce sur HDFS