Calcul distribué avec MapReduce

- Qu'est-ce que MapReduce?

1. Un modèle de programmation simple pour traiter d'énormes ensembles de données dans une manière distribuée.
2. Un framework qui exécute ces programmes sur des clusters en gérant automatiquement les détails de l'information distribuée:
    - ► Division du travail.
    - ► Distribution.
    - ► Synchronisation.
    - ► Tolérance aux pannes.

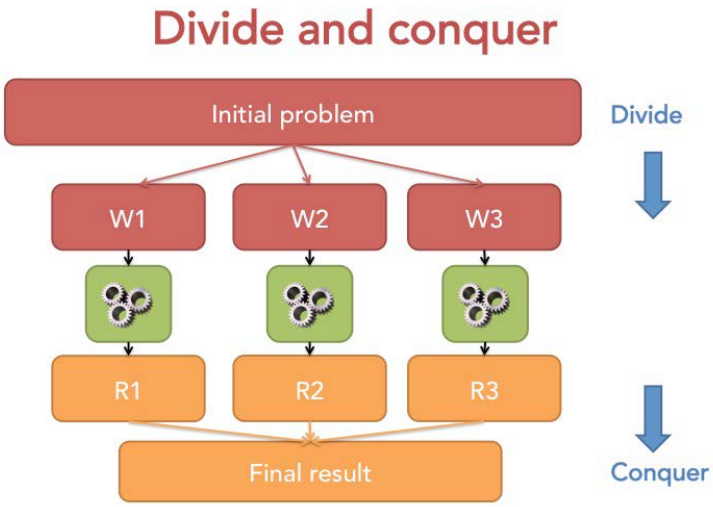
- MapReduce est un modèle de programmation et une implémentation associée pour traiter et générer de grands ensembles de données.
- Les utilisateurs spécifient une fonction de carte qui traite une paire **clé / valeur** pour générer un ensemble de paires clé / valeur intermédiaire, et une fonction de réduction qui fusionne toutes les valeurs intermédiaires associées avec la même clé intermédiaire.
- Les programmes écrits dans ce style fonctionnel sont automatiquement parallélisés et exécutés sur un grand groupe de machines de base.
- Le système d'exécution s'occupe des détails du partitionnement des données d'entrée, de la planification de l'exécution du programme sur un ensemble de machines et gestion des pannes de machines, et la communication inter-machine requise.
    - o Ceci permet aux programmeurs sans aucune expérience des systèmes parallèles et distribués d'utiliser facilement les ressources d'un grand système distribué.

Principe de diviser pour régner

Divisez une tâche en sous-tâches indépendantes.

Gérez les sous-tâches en parallèle.

Agréger les résultats des sous-tâches pour former la sortie finale



Problème typique de Big Data

- Itérer sur un grand nombre d'enregistrements.
- Extrayez les données de chaque enregistrement.
- Mélangez et triez (Shuffle and sort) les résultats intermédiaires.
- Agréger les résultats intermédiaires.
- Générez la sortie finale.

Idée clé de MapReduce :

- Le traitement de données peut être exprimé avec ces deux opérations (map et reduce)

  MAP

  - Itérer sur un grand nombre d'enregistrements.
  - Extraire des données de chaque enregistrement.
  - Mélanger et trier les résultats intermédiaires

  REDUCE

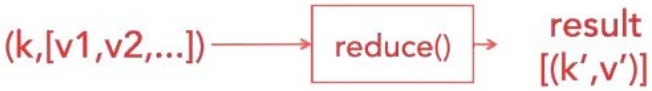
  - agréger les résultats intermédiaires
  - Générer la sortie finale.

Principes de base

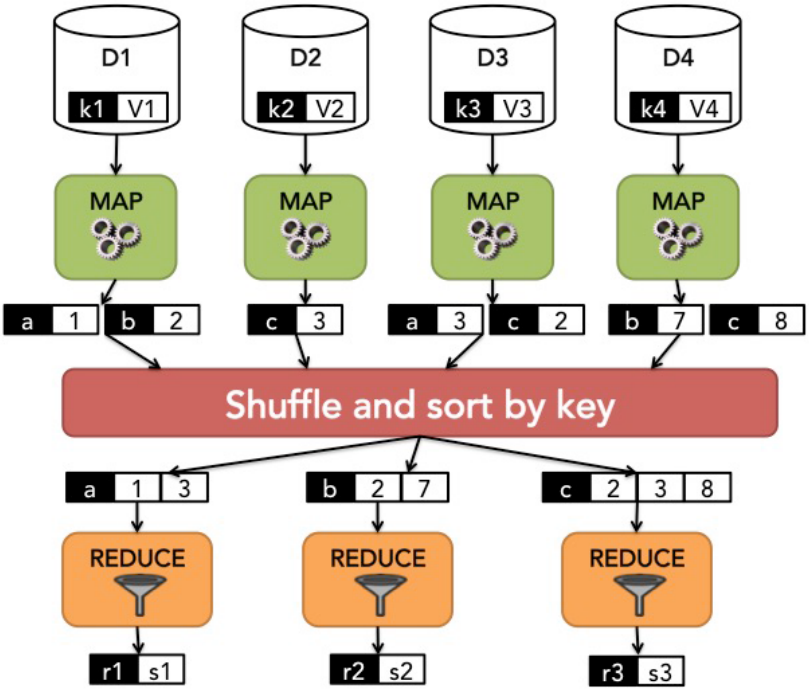
Toutes les données sont structurées en paires (clé, valeur).

Deux fonctions:

- MAP: transforme les données d'entrée en une liste de paires (clé, valeur).
- REDUCE: agréger / réduire toutes les valeurs associées à la même clé.



Présentation de MapReduce



Modèle de programmation MapReduce

Les programmeurs doivent spécifier deux fonctions:

- map $(k, v) \rightarrow [(k', v')...]$

▶ Prend une paire valeur / clé et génère un ensemble de paires valeur / clé

- par exemple **clé** = nom de fichier, **valeur** = le contenu du fichier

▶ Un appel de Map pour chaque paire (*K*, *V*).

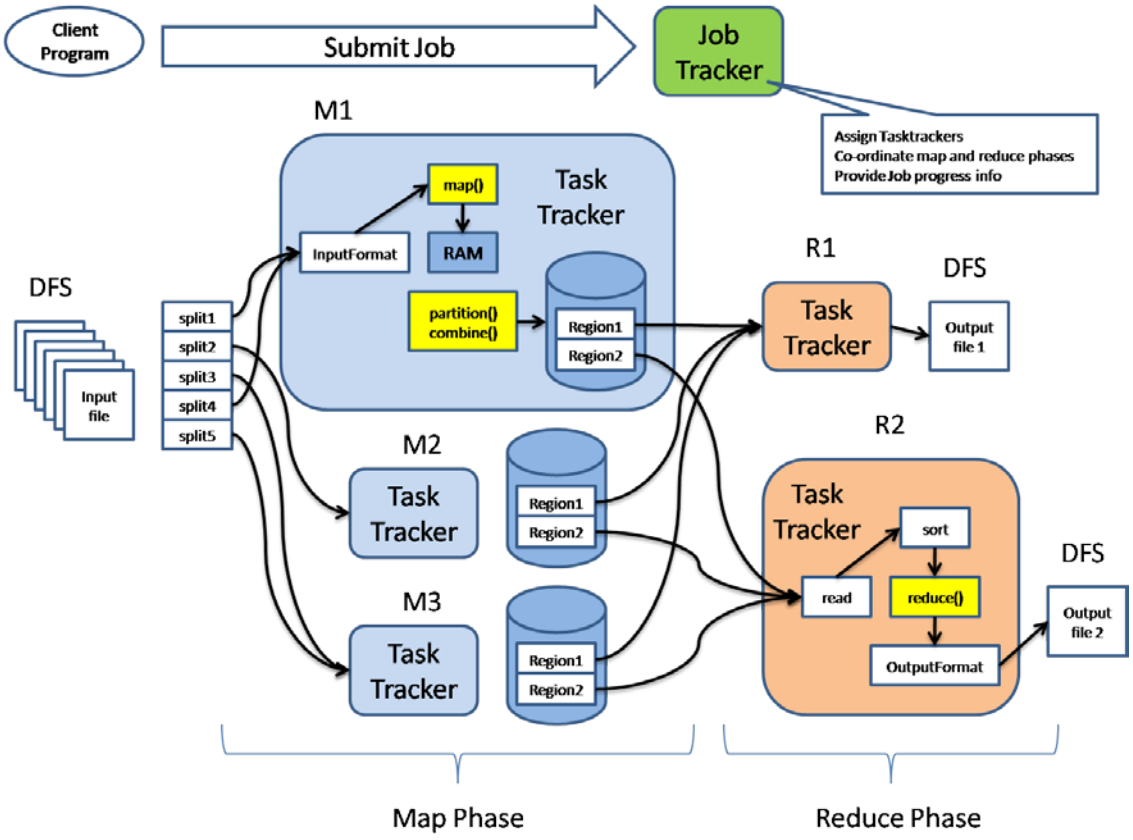
- reduce $(k', [v'_1, ...v'_n]) \rightarrow [(k'', v'')...]$

▶ Toutes les valeurs *V'* avec la même clé *K'* sont réduites ensemble et traitées dans l'ordre *V'*

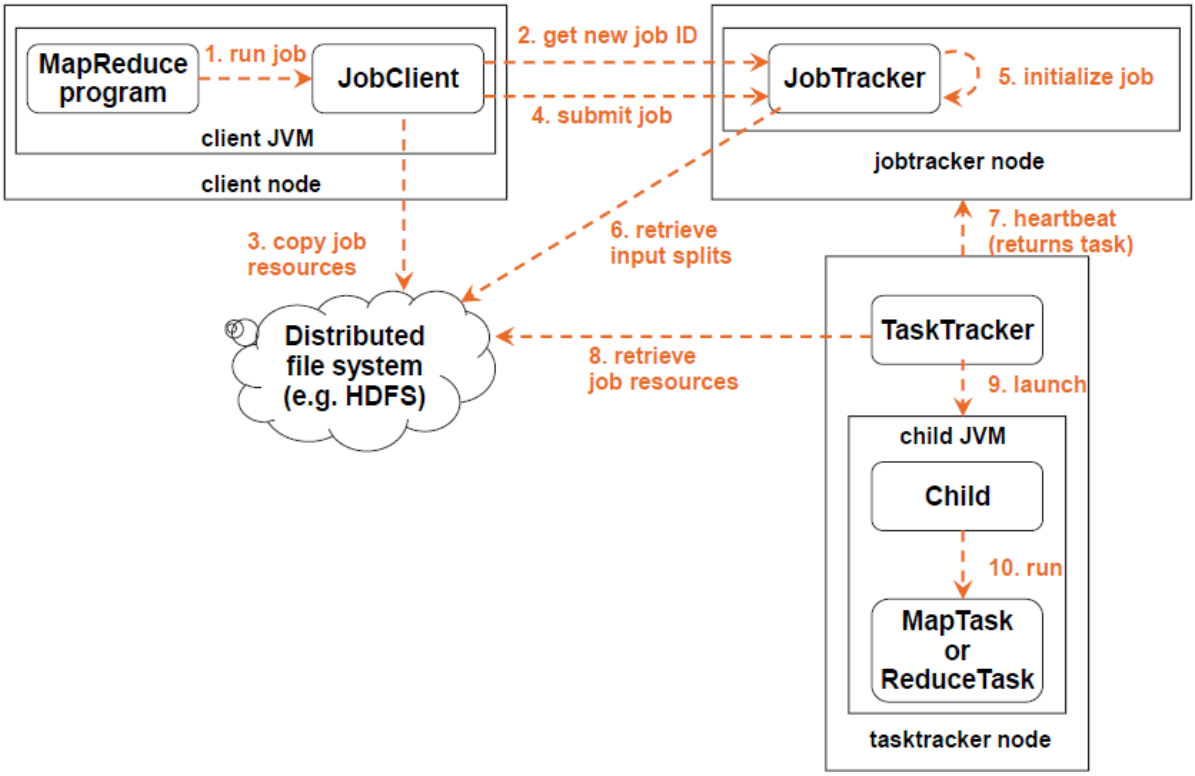
▶ Un appel de fonction Reduce pour chaque touche *K'*

L’architecture de MapReduce

- Le rôle principal du MapReduce consiste à **gérer les ressources** et à **guider le processus de calcul** (Job scheduling / monitoring).
- Le travail repose sur les phases « map » et « reduce », qui permettent aux fichiers d’être traités où ils sont stockés. Cela accélère le temps de calcul et minimise la consommation excessive de la bande passante du réseau.
  - Dans la phase du MAP, des processus de calculs complexes (jobs) sont divisés en unités et partagés par le **JobTracker** sur différents systèmes esclaves du cluster. Les **TaskTracker** veillent par la suite à ce que les divers processus partiels soient traités de manière parallèle.
  - Au cours de la Phase REDUCE, les résultats intermédiaires du MapReduce sont collectés et délivrent un résultat global.



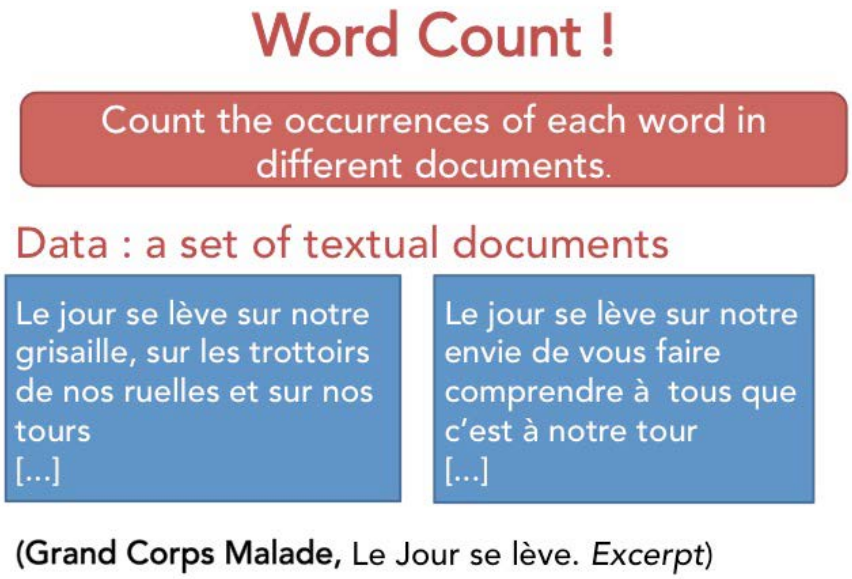
- Lors de l'exécution des programmes MapReduce, une distinction est faite entre le Job et la tâche.
  - *Job*: application complète pour le traitement des données dans le cluster
  - *Tâche*: exécution d'une seule étape de traitement sur un nœud de cluster

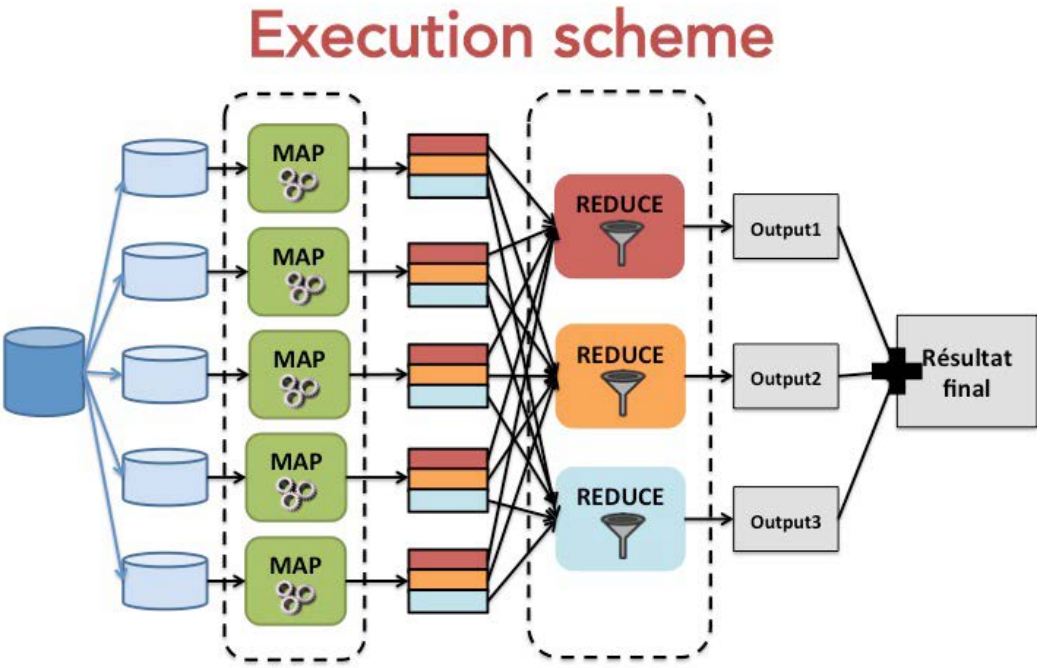


- Le rôle principal du JobTracker est de recevoir les Jobs du client, de les diviser en tâches et de transmettre les tâches au TaskTracker.
- Le JobTracker est situé sur un nœud maître et un démon qui contrôle les travaux MapReduce.
- JobTracker divise le Job en tâches individuelles et les distribue à d'autres nœuds de cluster (TaskTracker).
- Le JobTracker est responsable de
  - Gestion des ressources (gestion des nœuds TaskTracker),
  - Suivi de l'utilisation / disponibilité des ressources
  - Gestion du cycle de vie des Jobs (planification des tâches individuelles d'un Job, suivi de la progression, s'assurer que les tâches sont tolérantes aux erreurs, etc.)
- Le TaskTracker gère toutes les tâches qui s'exécutent sur le nœud de cluster.
  - Un démon TaskTracker s'exécute sur chaque nœud destiné au traitement.
  - Le TaskTracker est chargé d'instancier les tâches individuelles et de communiquer la progression de la tâche au JobTracker.
  - Le TaskTracker démarre et termine les tâches sur instruction du JobTracker et transmet régulièrement des informations sur l'état des tâches au JobTracker.

Exemple typique Word Count !

- Compte le nombre de fois où chaque mot distinct apparaît dans différents fichiers.





SPLIT (with simplification)

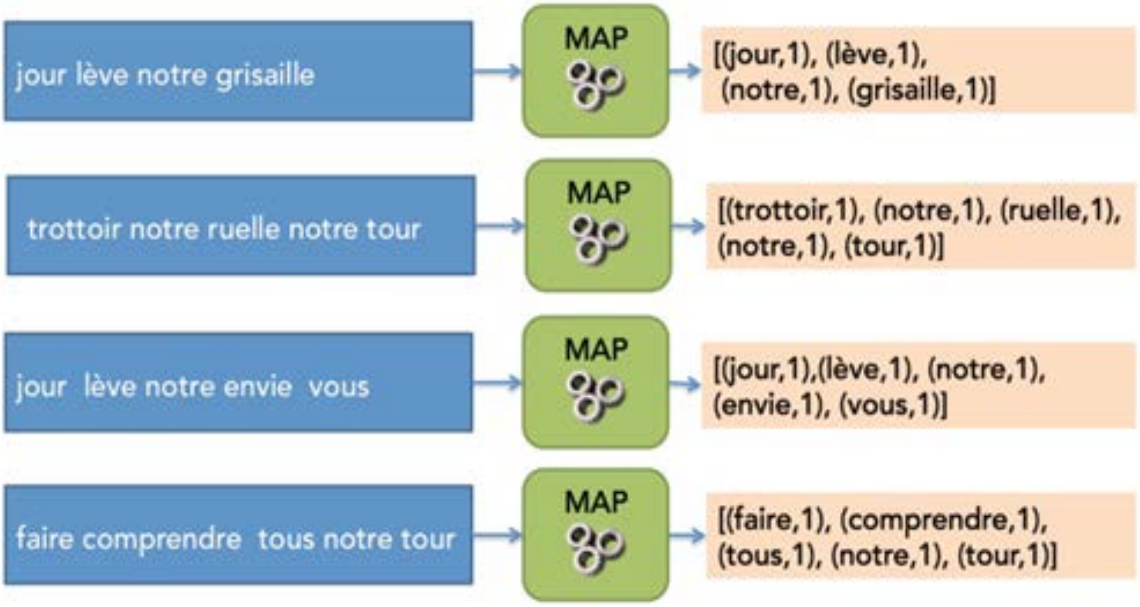
jour lève notre grisaille

trottoir notre ruelle notre tour

jour lève notre envie vous

faire comprendre tous notre tour

MAP



MAP

For each word, generate the pair (word, 1)

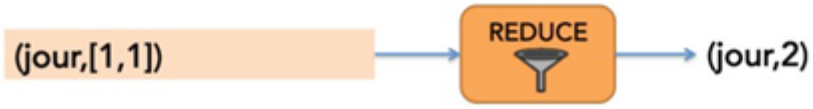
```
def map (key,value):     #key : nom du doc, value : contenu du doc
for word w in value:
    emitIntermediate (w,1)
```

SHUFFLE : group and sort all the pairs by key

(comprendre, [1])	(notre, [1,1,1,1,1])
(envie,[1])	(ruelle,[1])
(faire,[1])	(tour,[1,1])
(grisaille,[1])	(tous,[1])
(jour,[1,1])	(trottoir,[1])
(lève, [1,1])	(vous, [1])

REDUCE : add all the values associated to a same key

```
def  reduce (key,values):     #key : un mot, values : liste de valeurs
result =0
for count c in values:
     result = result +1
emit(key,result)
```



Word Count en MapReduce