Map-Reduce : un paradigme de programmation pour le Big Data

Jonathan Lejeune

UPMC/LIP6-INRIA

CODEL – Master 2 SAR 2017/2018

Motivations

Constat

2,5 ExaOctets produites par jour *(source IBM)* :

- scientifiques : capteurs, télescopes, accélérateur de particules, ...
- réseaux sociaux : messages, photos, vidéos ...
- commerciales : transactions d'achat, trading, ...

Les 4 V du BIG DATA

- **Volume** : traiter et stocker une grande masse de données
- **Vélocité** : vitesse très élevée de génération des données
- **Variété** : données structurées/non structurées sur des formats variés
- **Véracité** : assurer l'intégrité des données (obsolescence, justesse, ...)

Le Map-Reduce

Conçu par Google et présenté à la conférence OSDI 2004

Un paradigme de programmation

- Un flux de données basé sur la lecture et la production de clé/valeur
- 2 fonctions à programmer : **map** et **reduce**

Un environnement d'exécution

- Automatisation de la parallélisation sur un ensemble d'unités de calcul :
  - distribution des traitements
  - distribution des données
- Équilibrage de charge
- Stockage et transfert de données
- Gestion des éventuelles fautes et de la communication entre machines

$\Longrightarrow$ **transparent pour le programmeur**

Flux de données global du Map-Reduce

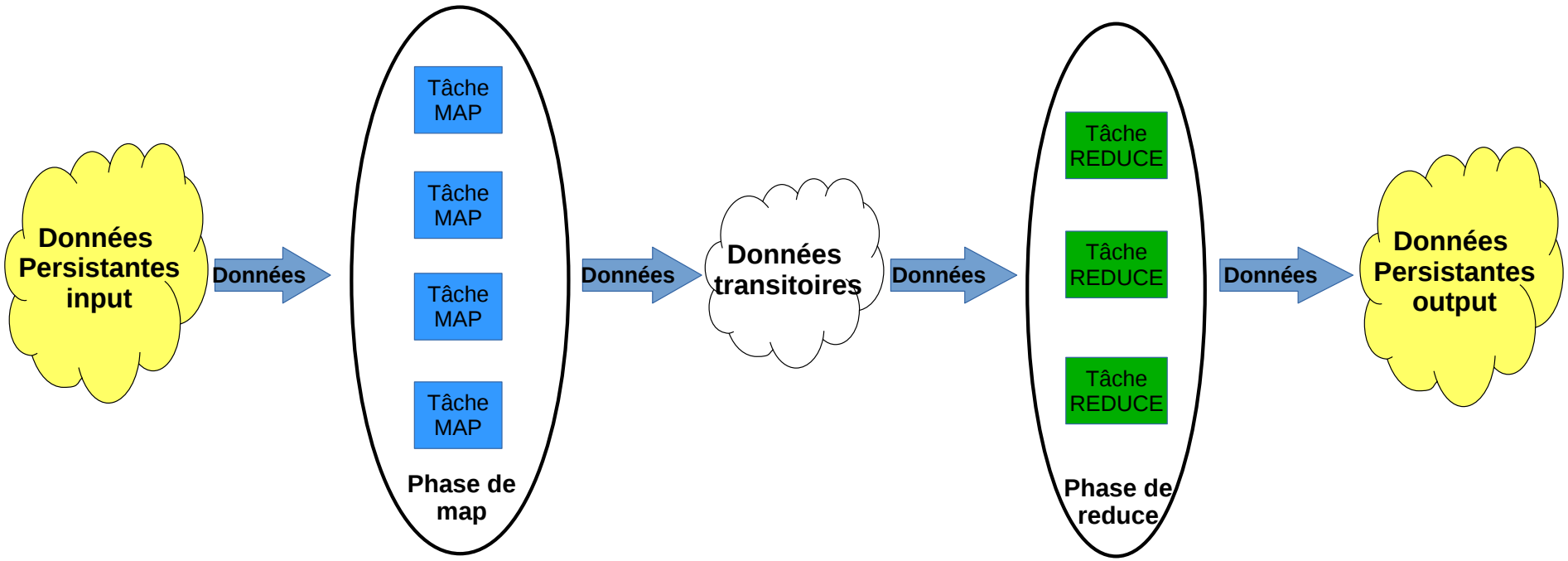


Schéma d'un calcul Map-Reduce

La phase de map

- Lit les données d'entrée sous la forme de <clé,valeur>
- Fait un traitement (exemple : extraire une information précise)
- Produit des données de sortie sous la forme de <clé,valeur>

La phase intermédiaire (Shuffle)

- Transfert, tri et fusionne les données entre les maps et les reduces

La phase de reduce

- Lit les données des maps via le shuffle sous la forme <clé,valeur>
- Fait un traitement (exemple : somme, groupement, filtre, ...)
- Produit les données de sortie du calcul sous la forme de <clé,valeur>

$\Longrightarrow$ Le programmeur doit au minimum fournir les fonctions map et reduce pour que son programme fonctionne

La phase de Map

Caractéristiques

- chaque tâche map traite sa partie des données d'entrée appelé (**split**)
- Chaque élément du split est associé à une clé de type K1
- A chaque clé de type K1 lue depuis le split, la tâche map correspondante fait un appel à la fonction *map()*.
- La fonction *map()* produit dans le flux d'information une liste de <clé,valeur> intermédiaire de type <K2,V2>
- nombre de tâches de map = nombre de splits

$$\textbf{Map : (K1,V1)} \rightarrow \textbf{list(K2,V2)}$$

La phase de Reduce

Caractéristiques

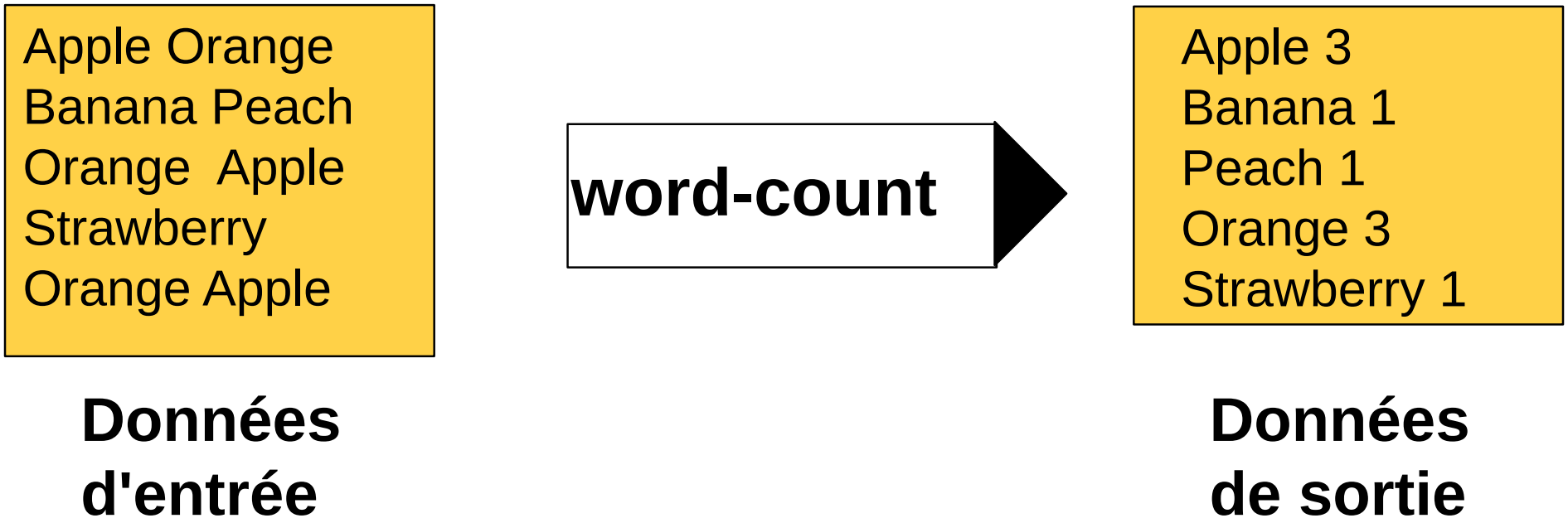
- Une fois la phase de map terminée, agrégation en liste de toutes les valeurs intermédiaires de type V2 associées à une clé de type K2.
- A chaque clé de type K2 la tache reduce correspondante fait un appel à la fonction *reduce()*.
- La fonction reduce() produit dans le flux d'information une liste de <clé,valeur> de type <K3,V3>
- Chaque paire <K3,V3> émise est enregistrée dans l'ensemble de données de sortie
- Le nombre de tâche de reduces est défini a priori par l'utilisateur

$$\textbf{Reduce : (K2, list(V2))} \rightarrow \textbf{list(K3, V3)}$$

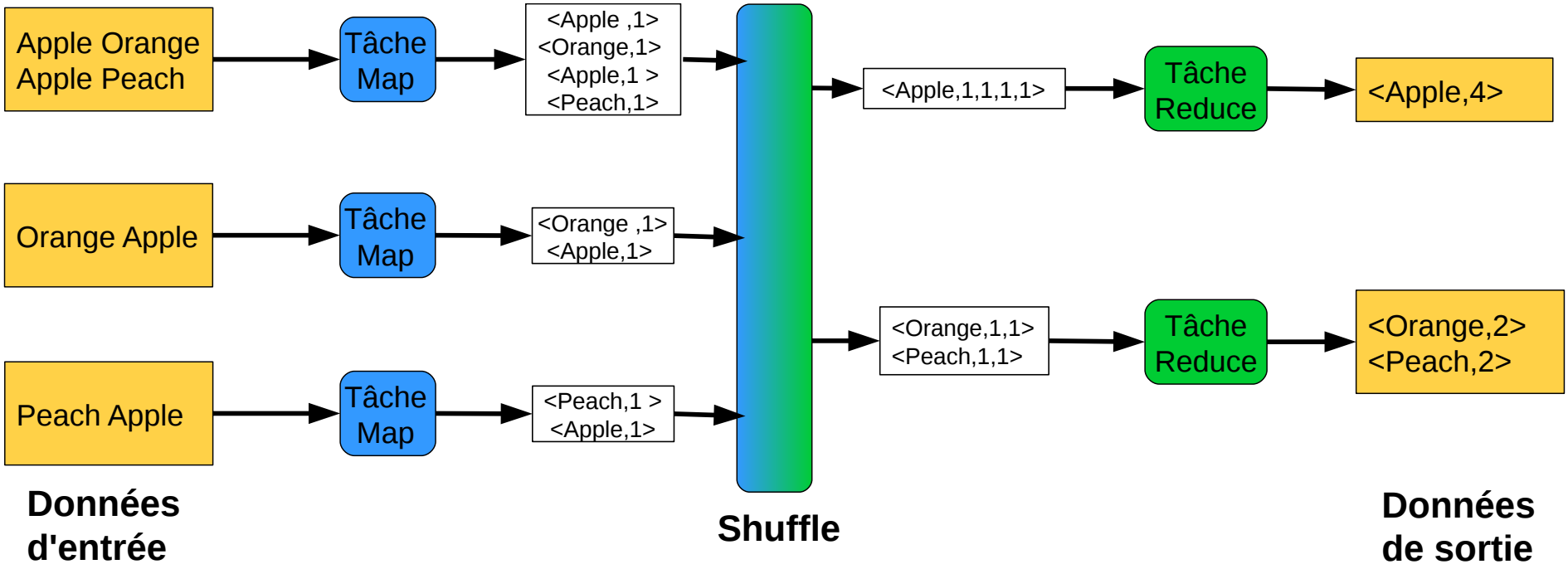
Remarque : bien souvent K2 = K3

Exemple de calcul : le word-count

- **En entrée** : un ou plusieurs (gros) fichiers textes
- **En sortie** : le nombre d'occurrences de chaque mot du texte



Exemple de flux de données avec le wordcount



Les fonctions Map et Reduce du wordcount

```
void Map(integer key, string value) {
     //key : id of the line
     //value : content of the line
       for each word w in value {
          Emit(w, 1);
       }
}

void Reduce(string key, list of integer values) {
     //key : a word
     //value : a list of counter
     integer count = 0;
     for each v in values {
          count +=  v;
     }
     Emit(key, count);
}
```

La phase de Shuffle

Transmettre des données de la phase map vers la phase reduce

Responsabilité des maps

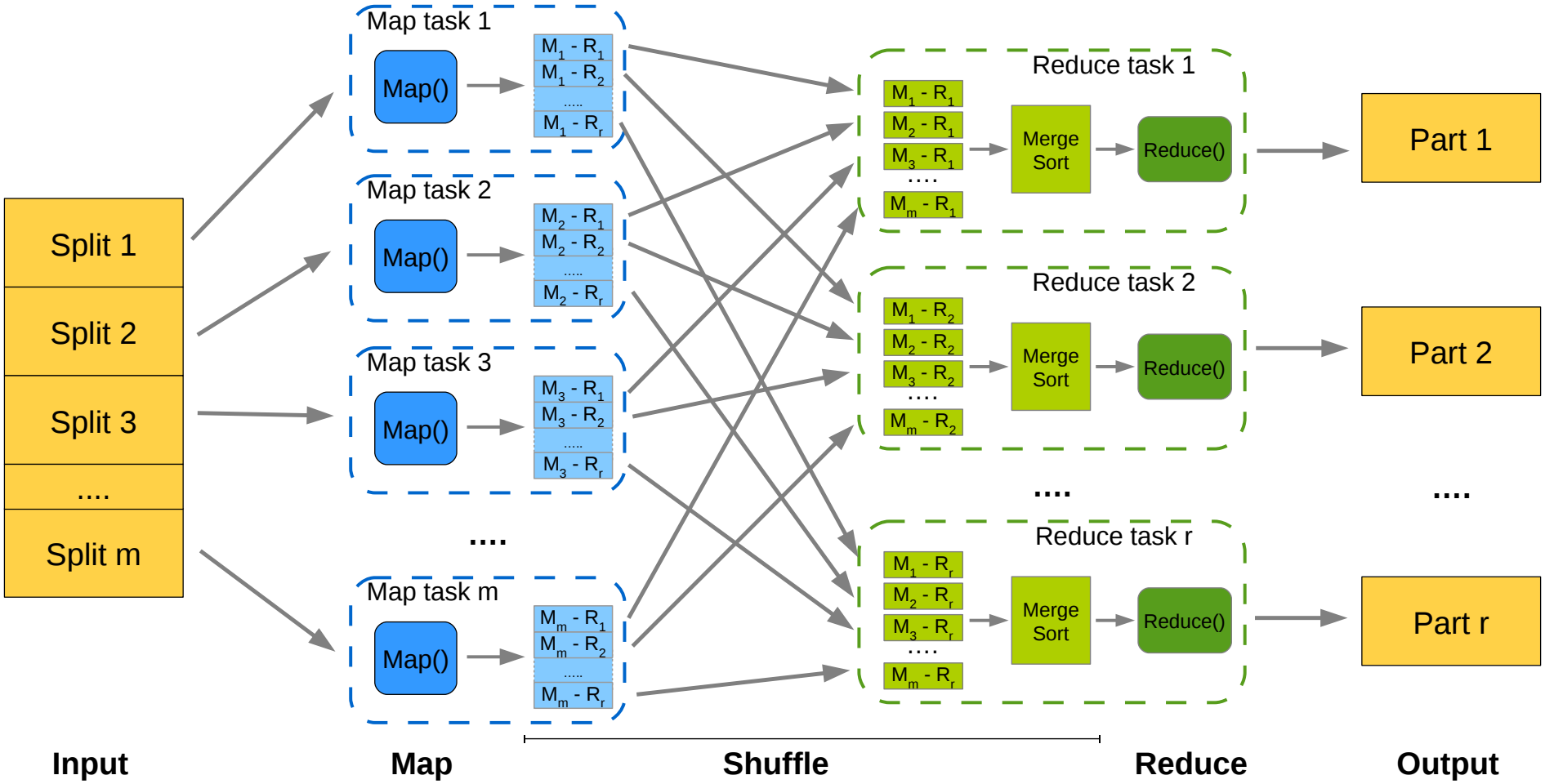
- stockage local partitionné des couples clé/valeur de sortie de map
- **Partitionnement :** assignement déterministe des clés sur *NbReduce* partitions
  $\Rightarrow$ une valeur de clé est associée à un unique reduce

Responsabilité des reduces

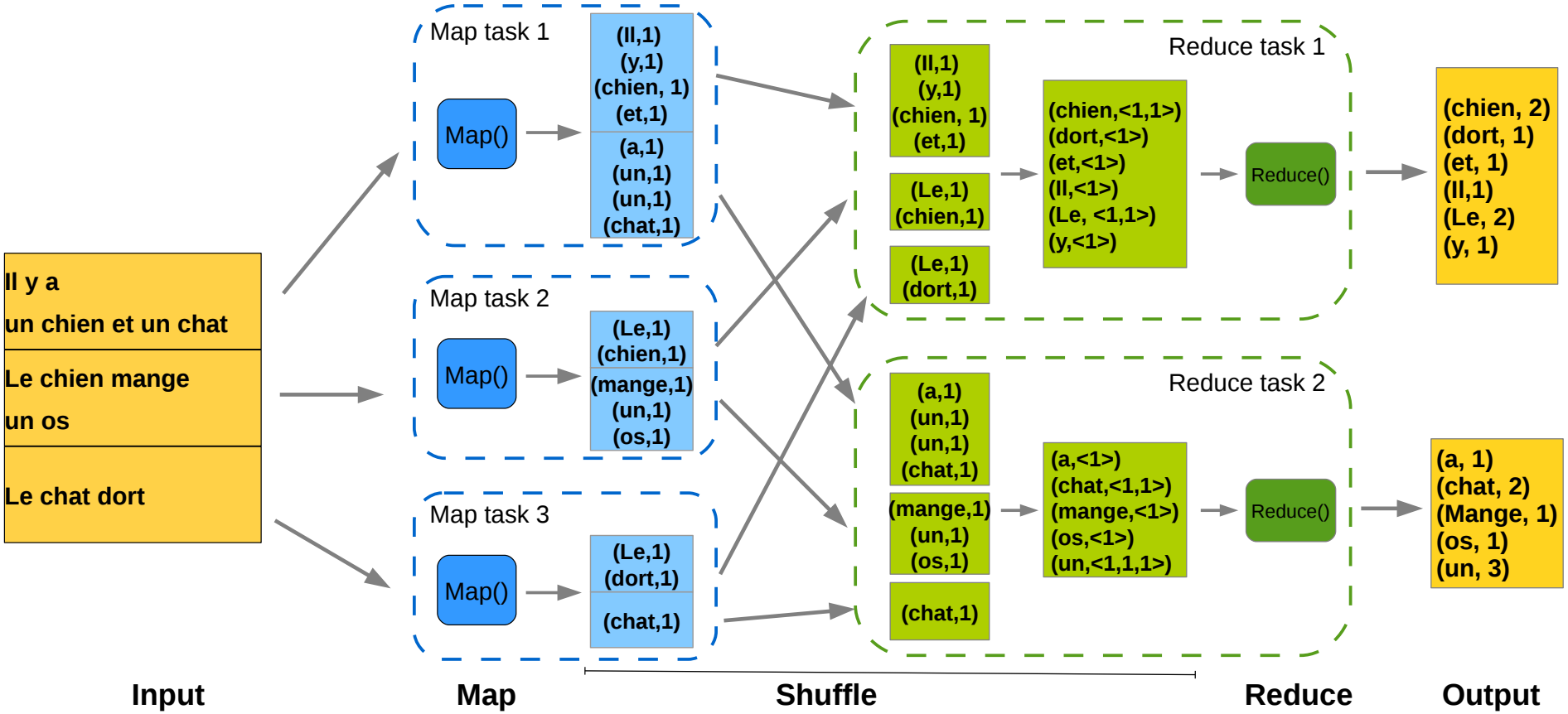
- **copy**
  - téléchargement sur chaque map de la partition qui lui est associée
- **merge**
  - agrégation de l'ensemble des partitions téléchargées
  - agrégation des valeurs pour une clé donnée
- **sort**
  - tri des différentes clés définissant l'ordre de lecture par le reduce : $\Rightarrow$ **un ordre doit être défini pour chaque type de clé**

Flux de données détaillé

IMPORTANT



Flux de données détaillé du WordCount



Bilan du programmeur Map Reduce

Le programmeur doit fournir à l'environnement d'exécution :

- une fonction de map
- une fonction de reduce
- des données

Le programmeur peut modifier :

- la politique de partitionnement du shuffle
- la politique de tri du shuffle
- le nombre de reduce
- la politique de découpage des splits
- les formats d'entrée et de sortie
- ...

SGBD vs. Map-Reduce

	SGBD	**Map-Reduce**
Taille	GigaOctets	PetaOctets
Accès	Interactif et batch	batch
Mises à jour	Plusieurs lectures et écritures	Une seule écriture, plusieurs lectures
Données structurées	oui	non
Intégrité	forte	faible
Passage à l'échelle	non linéaire	linéaire

Environnement d'exécution

Plusieurs implémentations :

- Google MapReduce :en C++, propriétaire, API Python et Java
- **Apache Hadoop** : en Java, open-source, API Java ou streams pour tout langage
- Amazon Elastic MapReduce : pour le cloud Amazon
- Microsoft HDInsight : pour le cloud Azur