Mast Desigeo - Cours BigData & NoSQL

Le Framework Map/Reduce

Auteurs : Cédric du Mouza et Nicolas Travers
Transparents repris de : Raphaël Fournier-S'niehotta, Philippe Rigaux, Nicolas Travers (Cours NFE204)

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

le cnam

Qu'est-ce que MapReduce

C'est un *framework* d'exécution.

- Un processus en deux phases (devinez ?) est appliqué à une collection d'items.
- À chaque phase, on applique une fonction fournie par le développeur.

Quel intérêt ?

- Implante de manière *générique* une exécution de type "group-by" à la SQL.
- **Scalable car facilement exécutable en parallèle**.

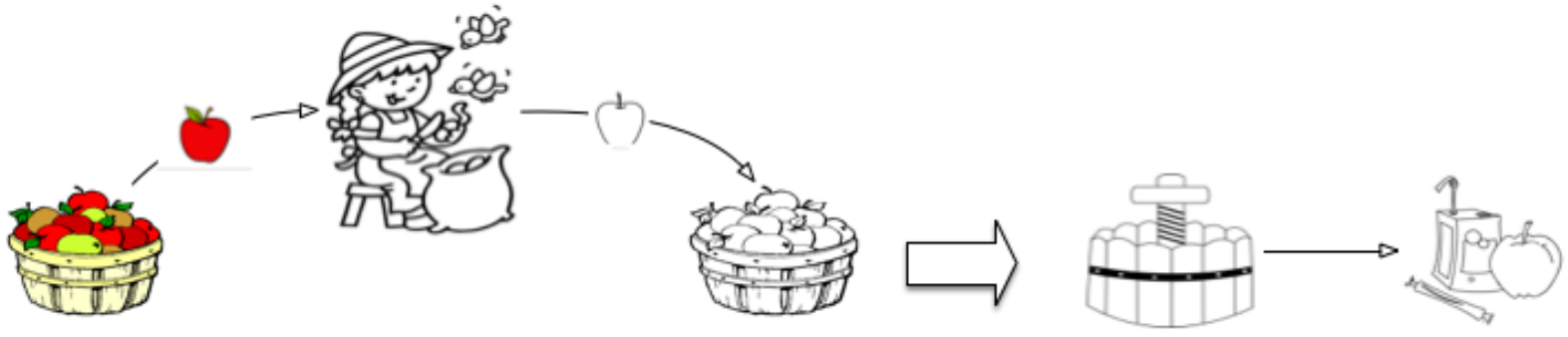
Ce n'est pas l'idéal !. Très contraint, très peu expressif, lent.

Cas particulier d'une **chaîne de traitement** scalable. Principe très ancien (prog. fonctionnelle), remis au goût du jour par Google (2004) pour traitements massivement distribués.

Les systèmes modernes ont étendu le principe : nous en reparlerons.

Faisons du jus de pomme avec MapReduce

Vous savez comment on fait du jus de pomme ? Voici comment on fait avec MapReduce (sans le dire).

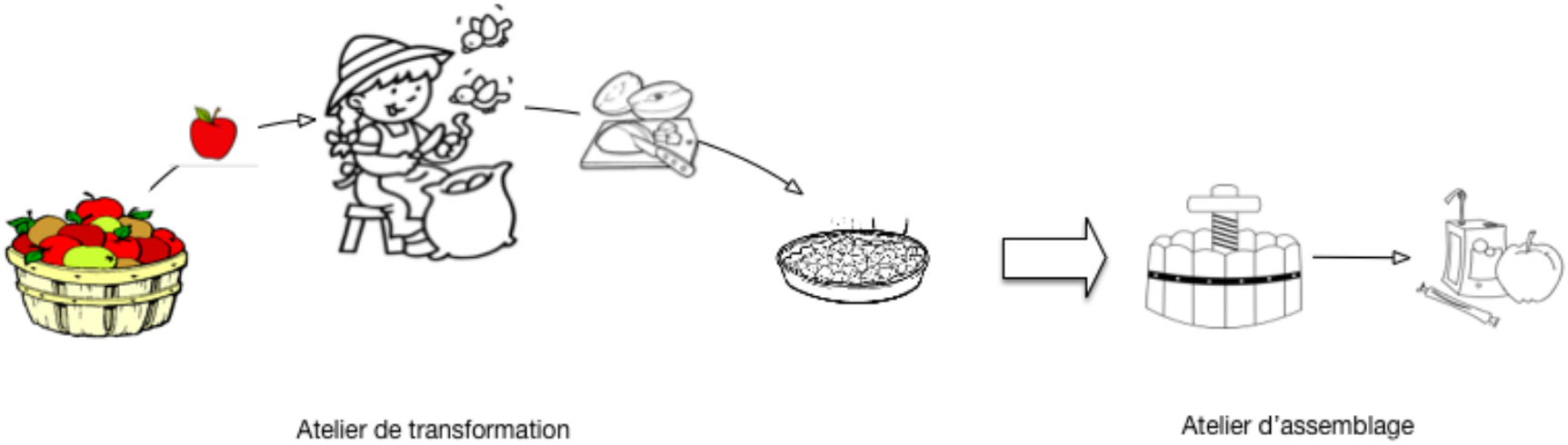


À retenir

- **L'atelier de préparation** : on transforme les items en entrée, un par un. **Traitement individuel, ordre indifférent.**
- **L'atelier d'assemblage** : on a **groupé** les items, on leur applique un **traitement collectif**.

Une petite extension

Rien n'impose d'avoir du 1 pour 1 dans l'atelier de préparation. On peut rejeter des pommes pourries, et couper les autres en quartiers.

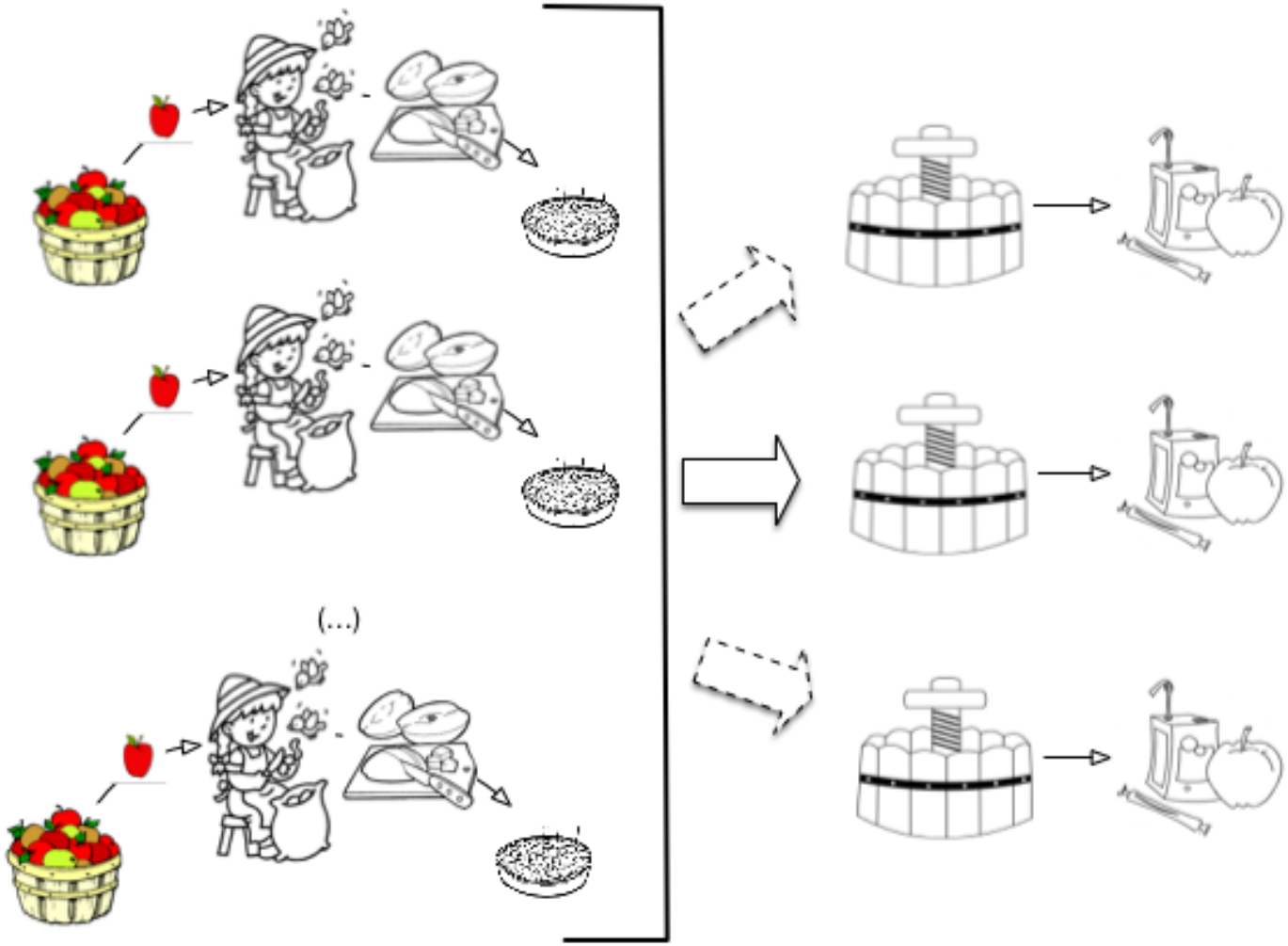


À retenir

L'atelier de préparation n'est pas limité à une transformation un pour un des produits consommés. Il peut prendre en entrée des produits d'une certaine nature, et sortir des produits d'une autre nature.

Passons au BigData

Si on veut faire beaucoup de jus de pomme : on parallélise.



Des remarques importantes

Parallélisation = indépendance des données et des transformations

Si l'ordre d'épluchage était important, ce ne serait pas si simple ;

Idem si l'épluchage d'une pomme dépendait de l'épluchage des autres.

Qu'est-ce qu'on gagne ?

n fois plus de ressources (cuisinier, matériel), n fois plus de jus de pomme !

Scalabilité

La production de jus de pomme est parallélisable et proportionnelle aux ressources (humaines et matérielles) affectées.

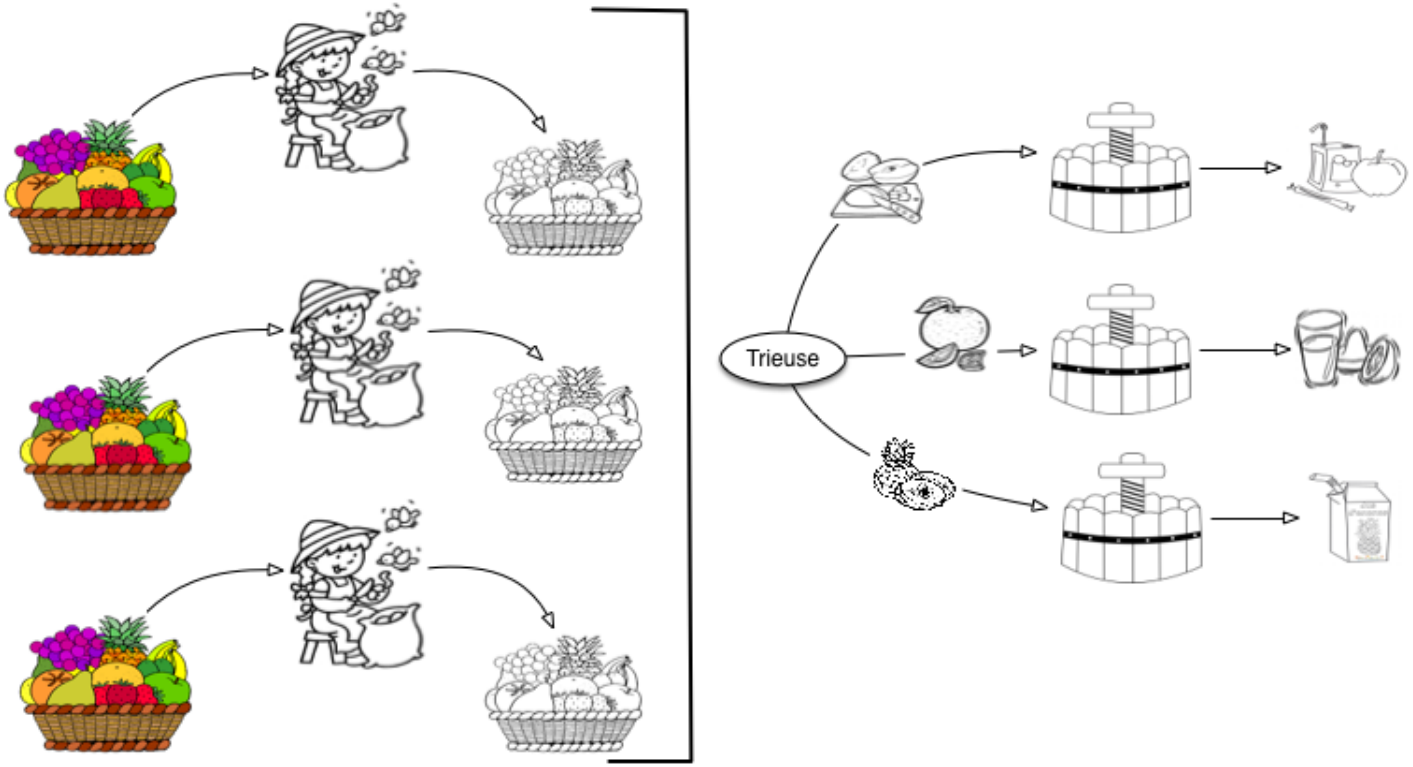
Mon processus est robuste !

Robustesse

Une panne affectant la production de jus de pomme n'a qu'un effet **local** et ne remet pas en cause **l'ensemble** de la production.

Encore mieux : je peux faire des jus de fruit

Jus de pomme, d'orange ou d'ananas : même procédé + un **tri**.



Encore des remarques importantes

Je dois trier et regrouper dans l'atelier d'assemblage

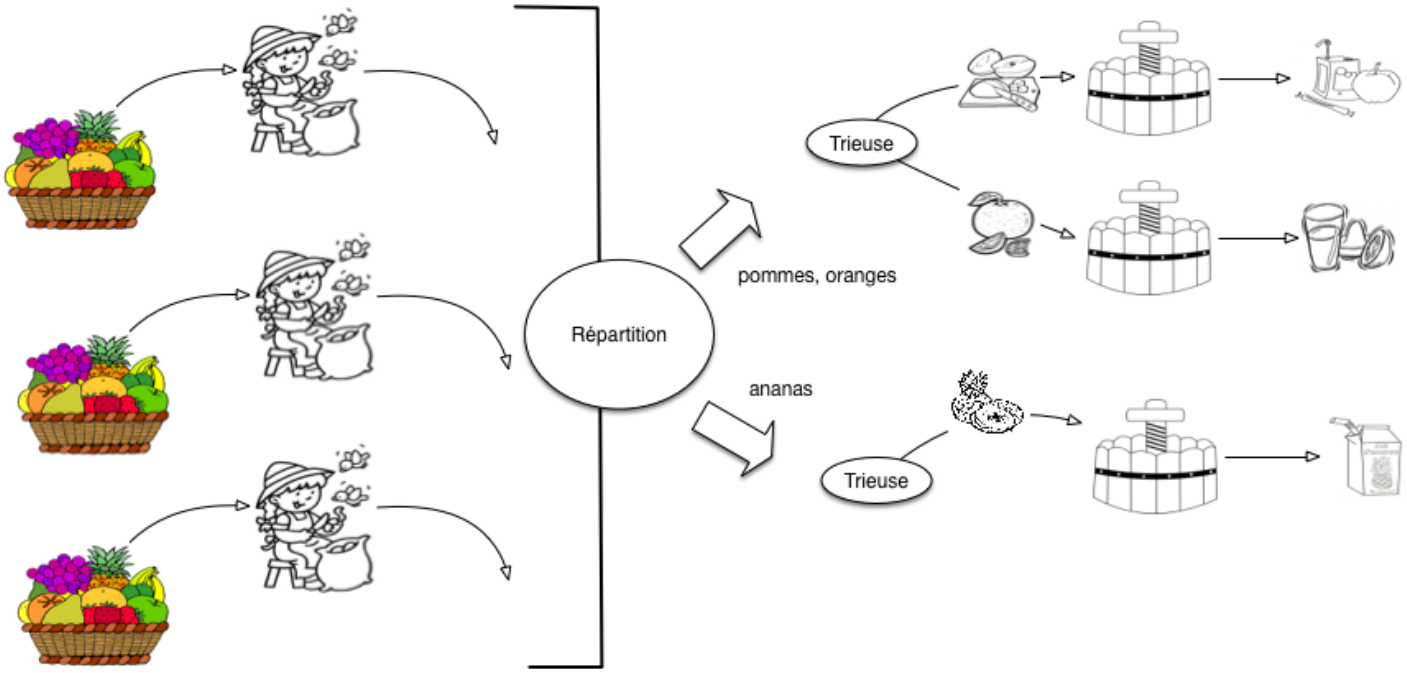
Ma transformation s'applique à des groupes distincts : ils sont constitués quand on initialise la phase d'assemblage.

Pour être capable de trier, je dois étiqueter les items

C'est le rôle de l'atelier de préparation : on met une étiquette sur chaque item avant de le transmettre à l'assemblage.

Bouquet final : je parallélise aussi l'assemblage

Et voici du Map Reduce complet.



Maintenant, je dois aussi **distribuer** les items : pommes et oranges en haut, ananas en bas.

Résumé (version basique)

La phase de `map` (préparation). Elle prend en paramètre une fonction $F_{map}()$.

- On parcourt la collection en entrée, document par document.
- On applique à chaque document d_i la fonction $F_{map}()$.
- On obtient une valeur v_i que l'on place dans un **accumulateur** A.

La phase de `reduce` (assemblage). Elle prend en paramètre une fonction $F_{red}()$.

- On applique $F_{red}()$ aux valeurs contenues dans l'accumulateur A.
- On obtient une valeur qui est le résultat du processus.

Essentiel

Les fonctions $F_{map}()$ et $F_{red}()$ sont **la partie "métier", fonctionnelle**. Le reste est une **infrastructure générique** prise en charge par un **framework**.

Dans notre schéma : le cuisinier est $F_{map}()$, le pressoir est $F_{red}()$

Notions et vocabulaire

À retenir :

- **Item (entrée)**, **document** en ce qui nous concerne.
- **Fonction de Map**, implante la transformation appliquée à un item pendant la phase de Map.
- **Paire intermédiaire**, c'est le résultat de la fonction de Map.
  Une paire intermédiaire (k, v) comprend l'identifiant (étiquette) du groupe auquel v appartient.
- **Groupe intermédiaire**, ensemble des valeurs produites par la fonction de Map et partageant le même identifiant de groupe.
- **Fonction de Reduce**, implante la transformation appliquée à un groupe pendant la phase de Reduce.

Ne pas retenir (pour l'instant) : comment tout cela s'exécute en distribué, avec parallélisation et reprise sur panne.

Vision d'ensemble (en centralisé)