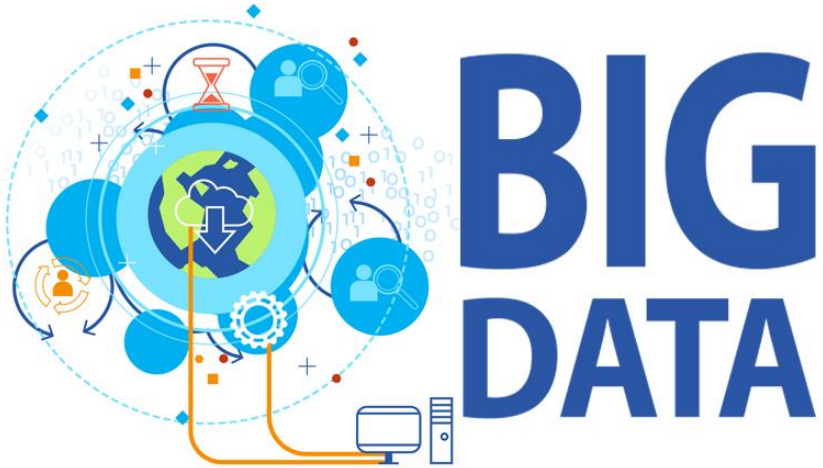


L'éco-Système HADOOP:

Framework: MapReduce

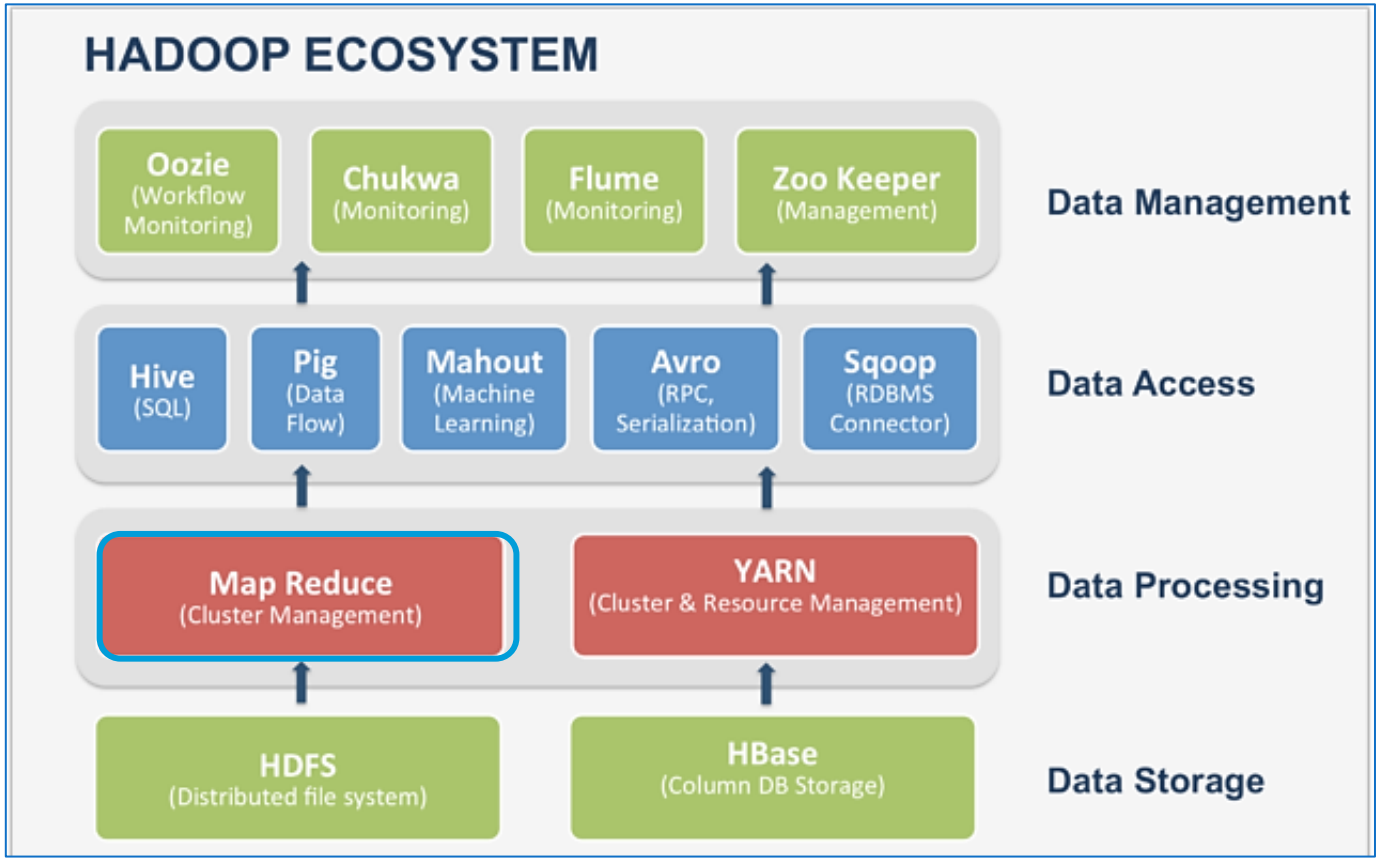
Pr. HIBBI Fatima-Zohra

f.hibbi@emsi.ma

Plan

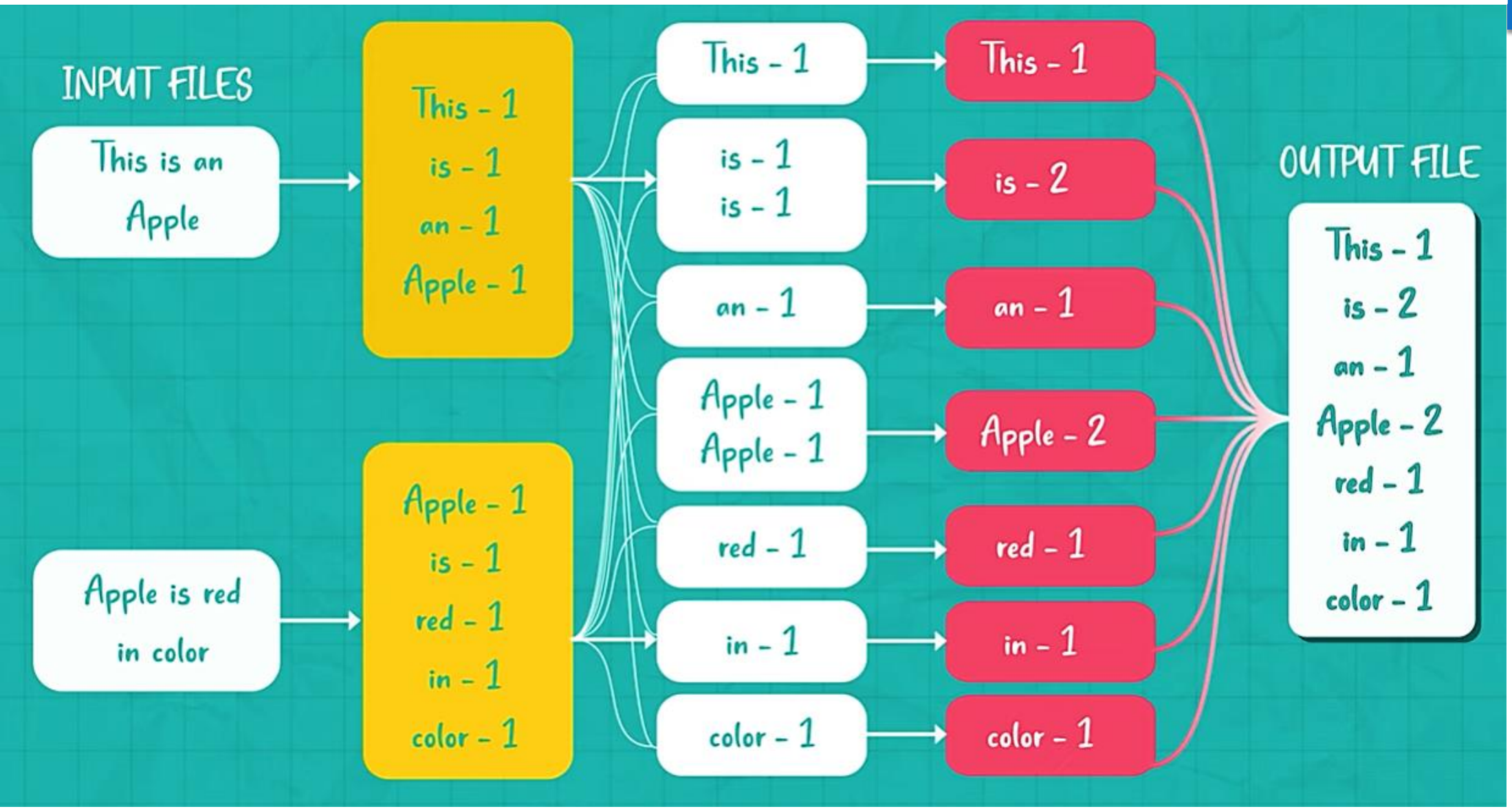
- Principe de MapReduce
- L'architecture de MapReduce
- Modèle de programmation: MapReduce
- MapReduce: Tolérance aux pannes
- YARN

Introduction



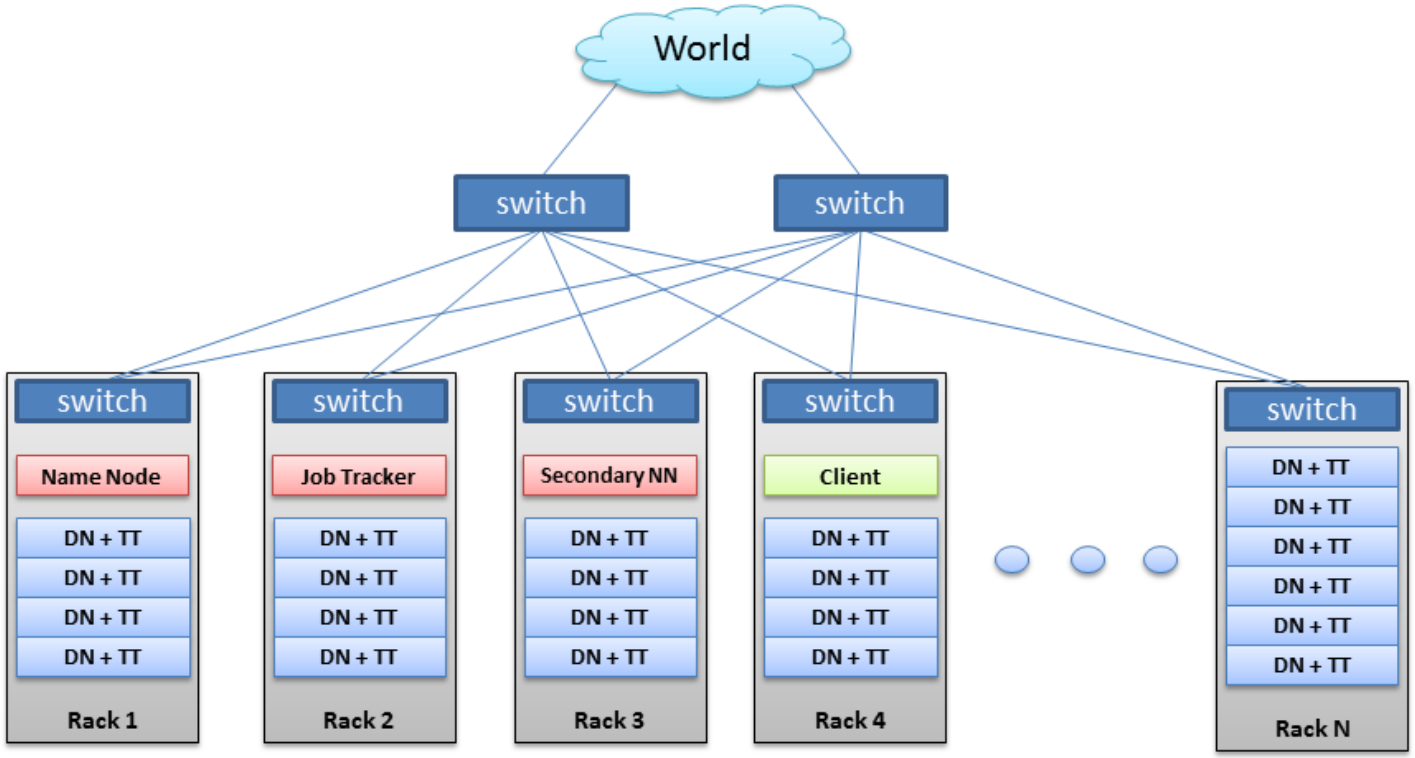
-MapReduce : un framework de traitement distribué;

-Il permet d'effectuer des calculs de façon parallèles.


INPUT FILES
This is an Apple
Apple is red in color
This - 1
is - 1
an - 1
Apple - 1
Apple - 1
is - 1
red - 1
in - 1
color - 1
This - 1
is - 1
is - 1
an - 1
Apple - 1
Apple - 1
red - 1
in - 1
color - 1
This - 1
is - 2
an - 1
Apple - 2
red - 1
in - 1
color - 1
OUTPUT FILE
This - 1
is - 2
an - 1
Apple - 2
red - 1
in - 1
color - 1

Framework: MapReduce

Hadoop cluster



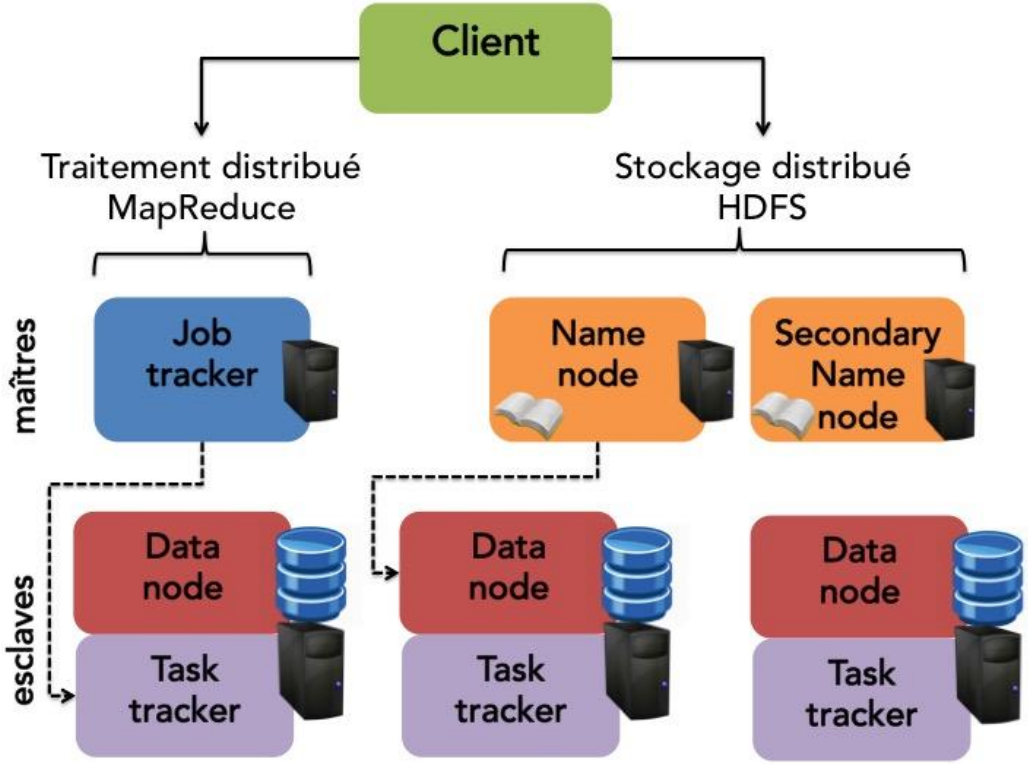
MapReduce: Principe

§ L'architecture de MR1:

-Maître/Esclave: L'unique maître (JobTracker) contrôle l'exécution des deux fonctions sur plusieurs esclaves (TaskTrackers).

§ Un programme MapReduce contient deux fonctions principales Map () et Reduce () contenant les traitements à appliquer au données.

§ Langage de programmation utilisé: Java, Python, etc.

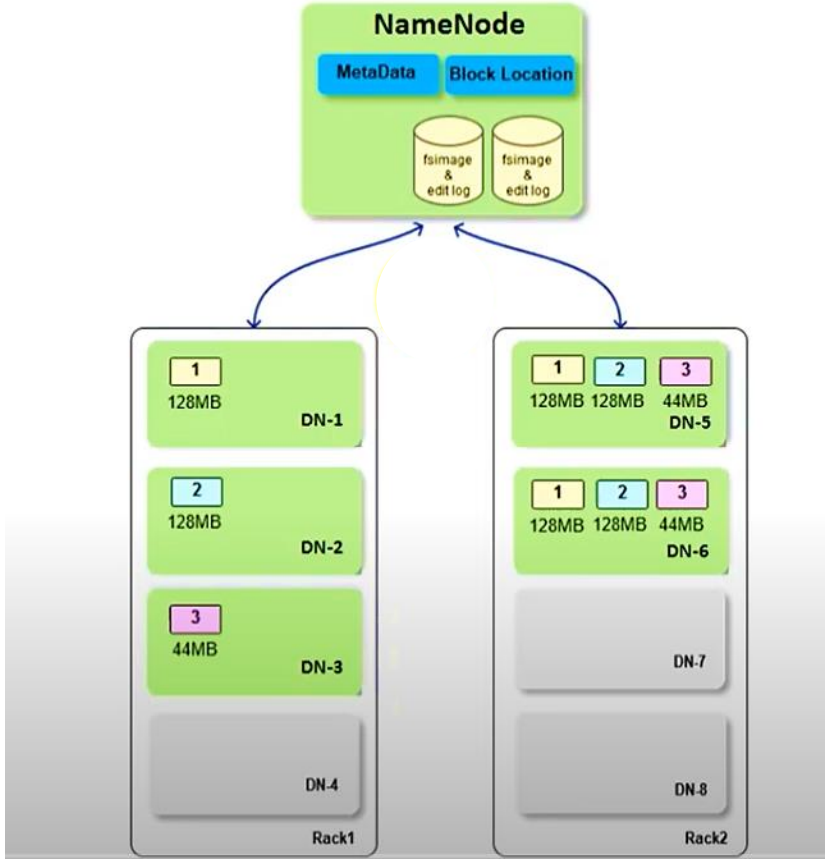


MapReduce: Principe

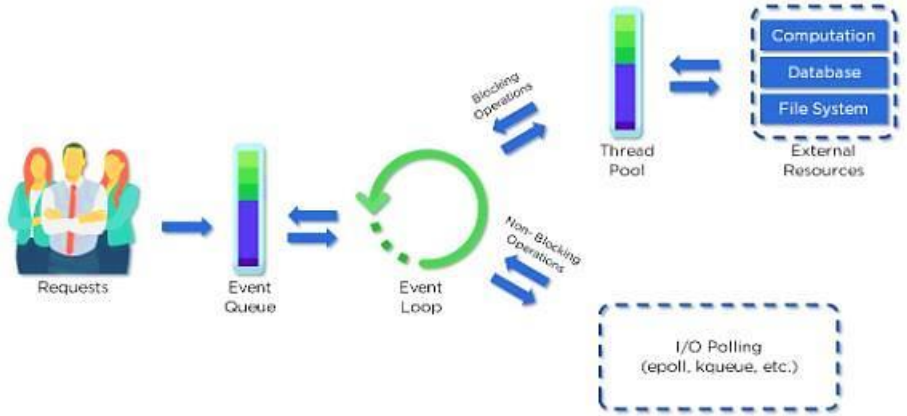
§ MapReduce résout le problème de traitement des données en:

-Divisant les taches de traitement en plus **petites parties** et en les assignant à plusieurs ordinateurs: Pour de meilleures performances, chaque **bloc** de données est traité **localement**(principe de **data locality**), minimisant les besoins d'échanges réseaux entre les machines.

-A la fin du traitement, les résultats sont collecté à un seul endroit et intégré pour former le résultat de traitement.



MapReduce: Architecture



MapReduce: JobTracker

-Reçoit les jobs MapReduce envoyés par les clients (Applications)

-Communique avec le NameNode pour avoir les emplacements des données à traiter

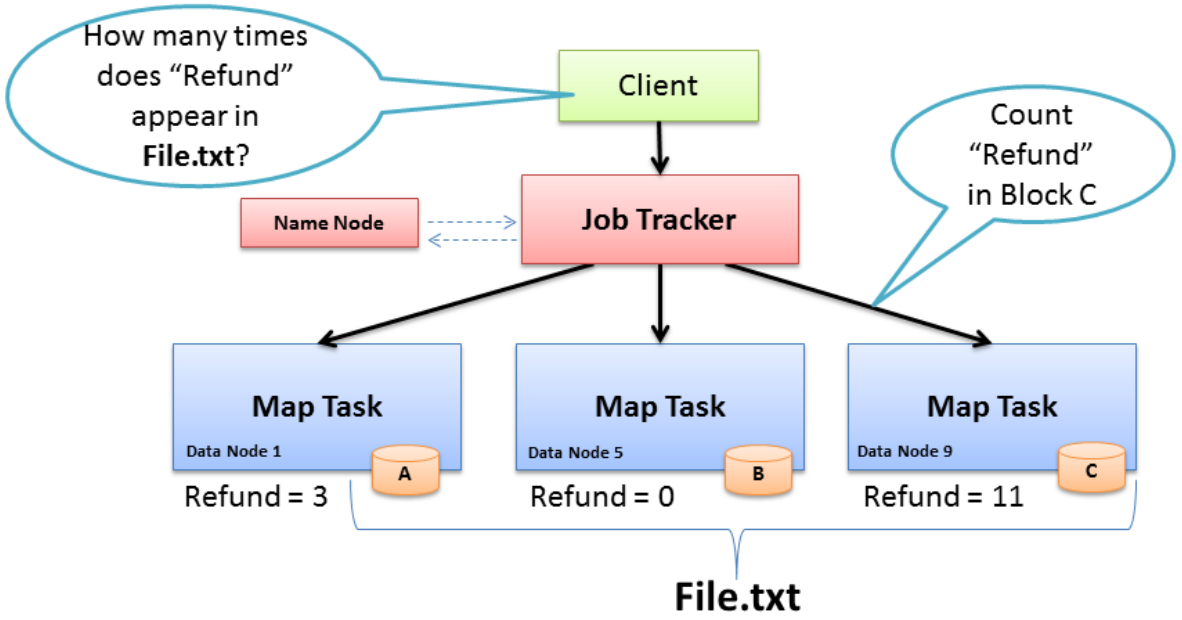
-Passe les tâches Map et Reduce aux nœuds TaskTrackers (**selon l'emplaçement des blocks**)

-Surveille les tâches et le statut des TaskTrackers

-Relance une tâche si elle échoue.

-Surveille l'état d'avancement des jobs et fournit des informations à ce sujet aux applications clientes.

-Algorithme d'ordonnancement des jobs est par défaut FIFO.



JobTracker fait: la gestion des ressources, l'ordonnancement et la surveillance des tâches

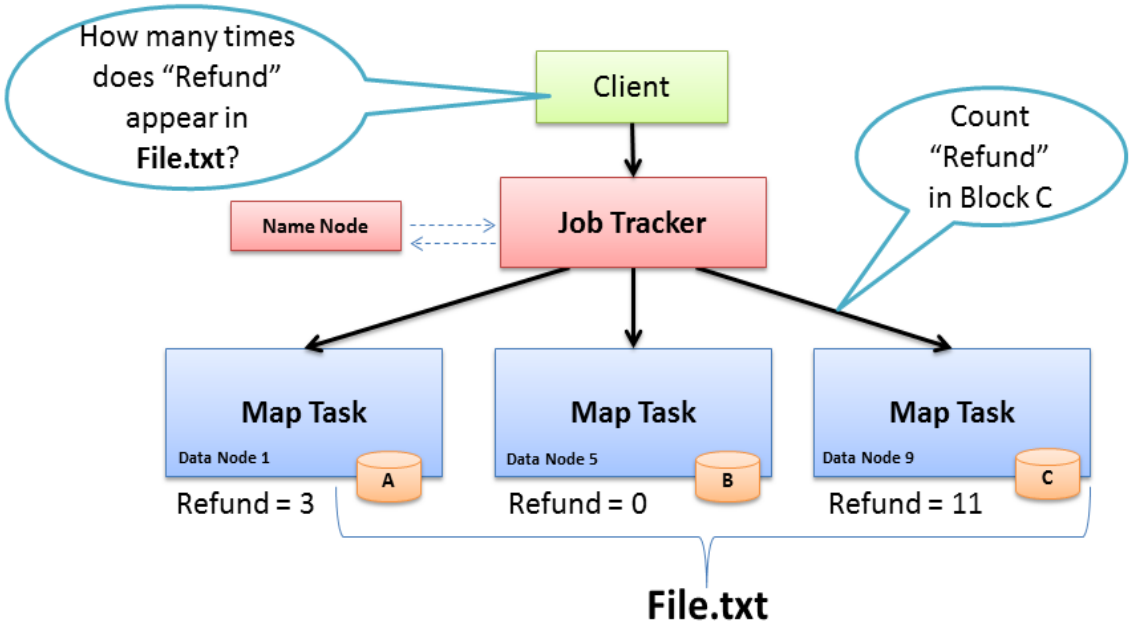
MapReduce: TaskTracker

-Exécute les tâches Map et Reduce

-Chaque TaskTracker possède un nombre de "slots" pour exécuter les tâches Map et Reduce (mapreduce.tasktracker.map.tasks.maximum et mapreduce.tasktracker.reduce.tasks.maximum dans mapred-site.xml).

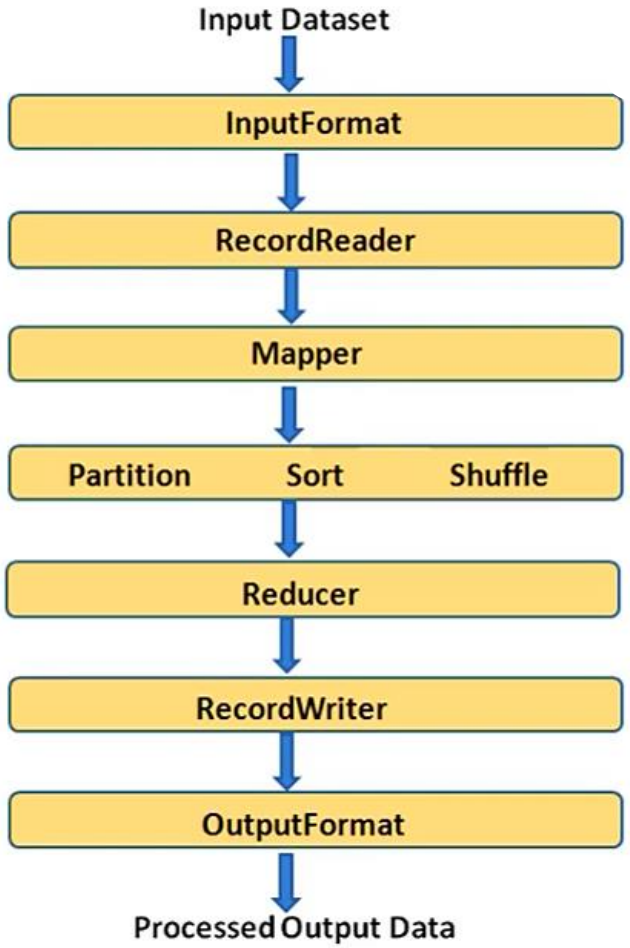
-Communique son statut au JobTracker via des heartbeat (slots libres, ...)

-Après une durée de mapreduce.jobtracker.expire.trackers.interval (10 minutes par défaut) le TaskTracker est considéré comme perdu.



TaskTracker fait: Gère le stockage des sorties intermédiaires (Stockage local)

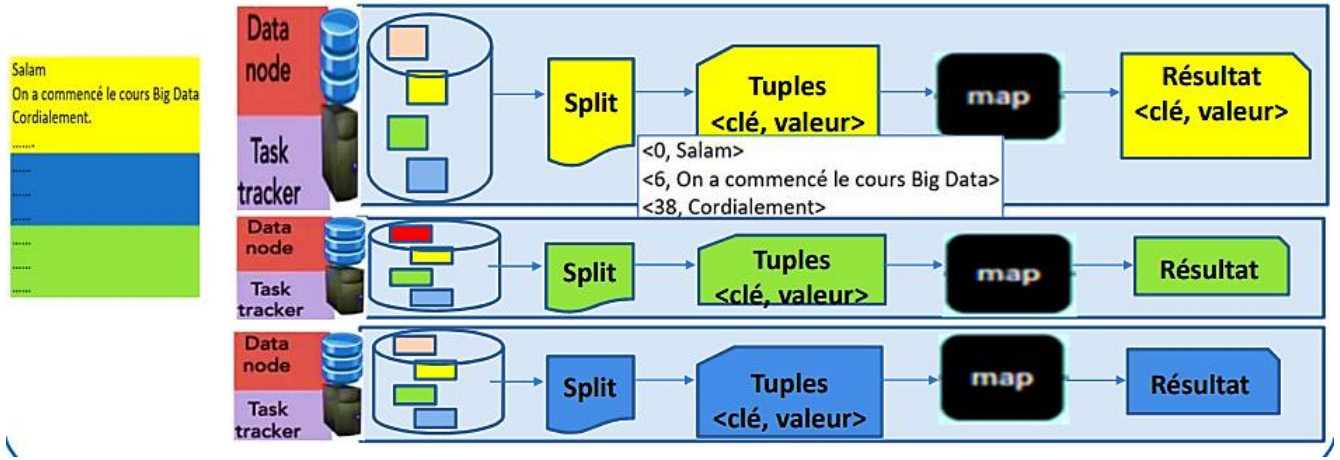
Modèle de Programmation: MapReduce



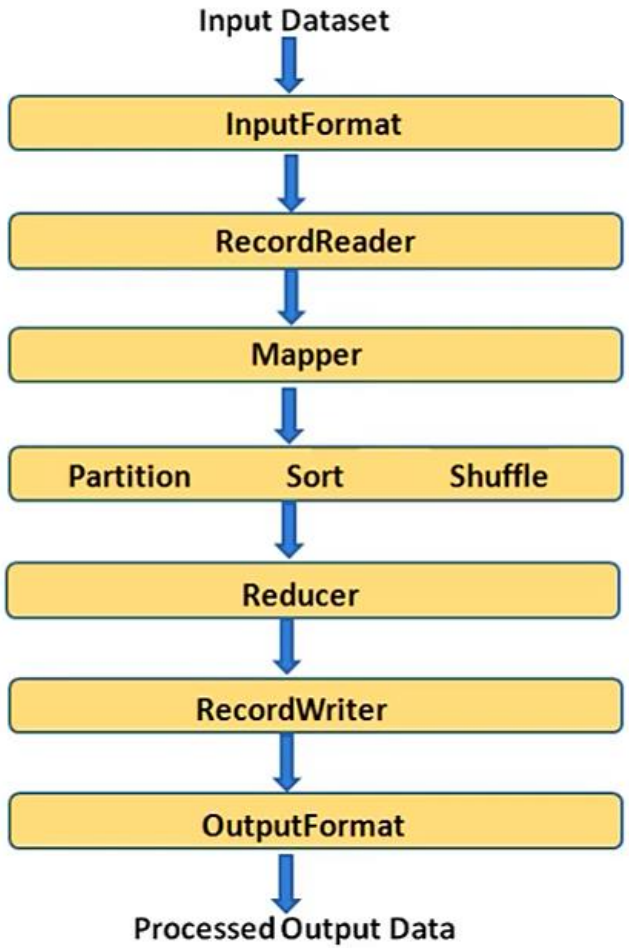
-Input Format : divise les fichiers à traiter en plusieurs **splits.** TextInputFormat est le format par défaut.

-RecordReader: Lecture des fichiers (splits) et la conversion des lignes en paires <Clé,Valeur>

-Mapper: traite les enregistrements provenant du RecordReader et génère les nouvelles paires <clé, valeur>.

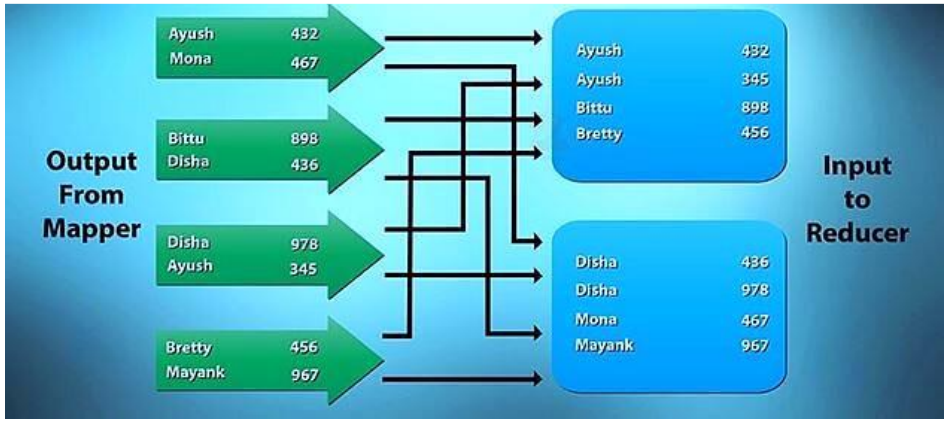


Modèle de Programmation: MapReduce

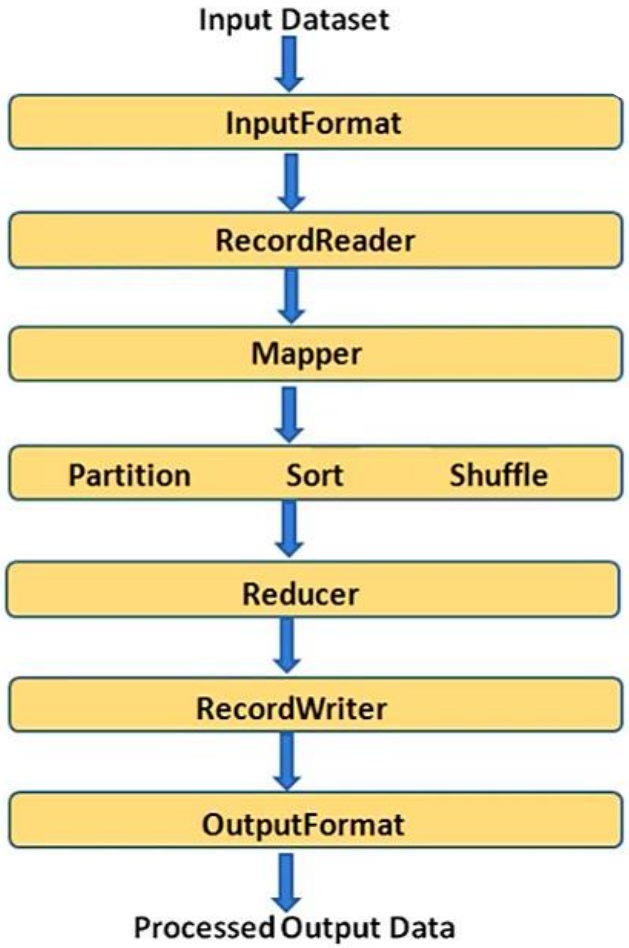


-Partition: La classe de partition détermine la partition dans laquelle une paire donnée (clé, valeur) ira

-Sort & Shuffle (Réorganisation): Le résultat ({<clé, valeur>}) produit par chaque Map est localement regroupée par clé.



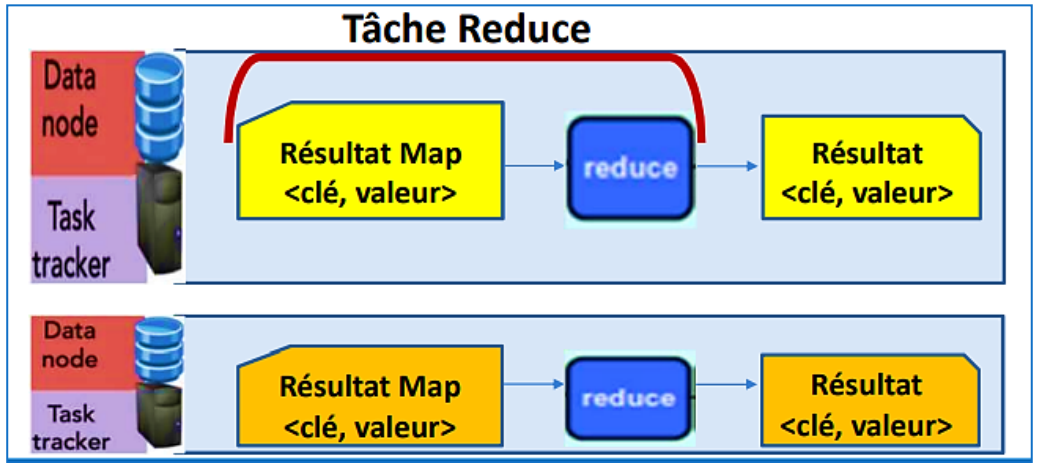
Modèle de Programmation: MapReduce



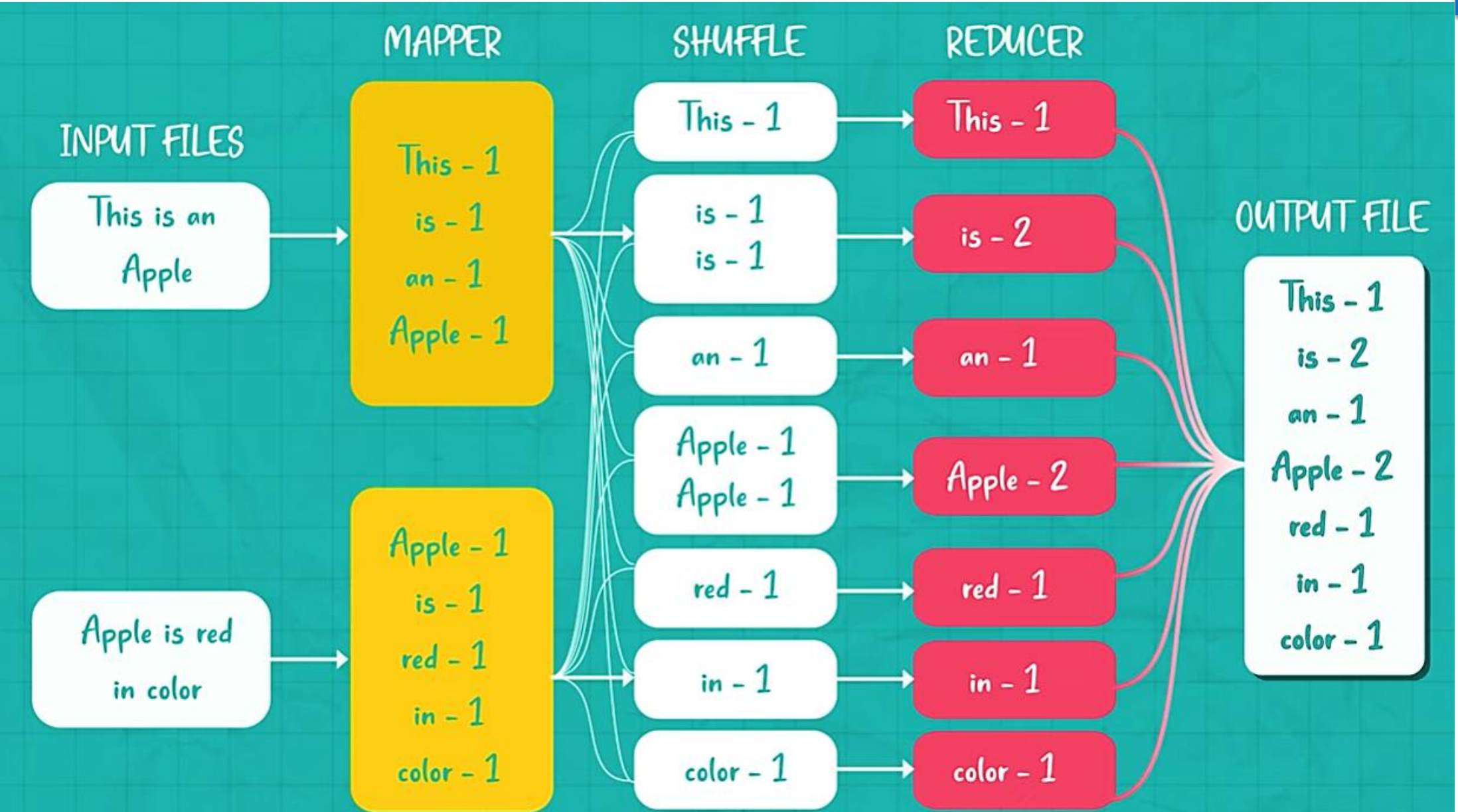
-Reducer: Petit programme (généralement) qui traite toutes les valeurs de la clé dont il est responsable. Ces valeurs sont passées au Reducer sous forme d'un tableau

-RecordWriter: Il écrit les résultats de la tâche MapReduce dans le système de fichiers

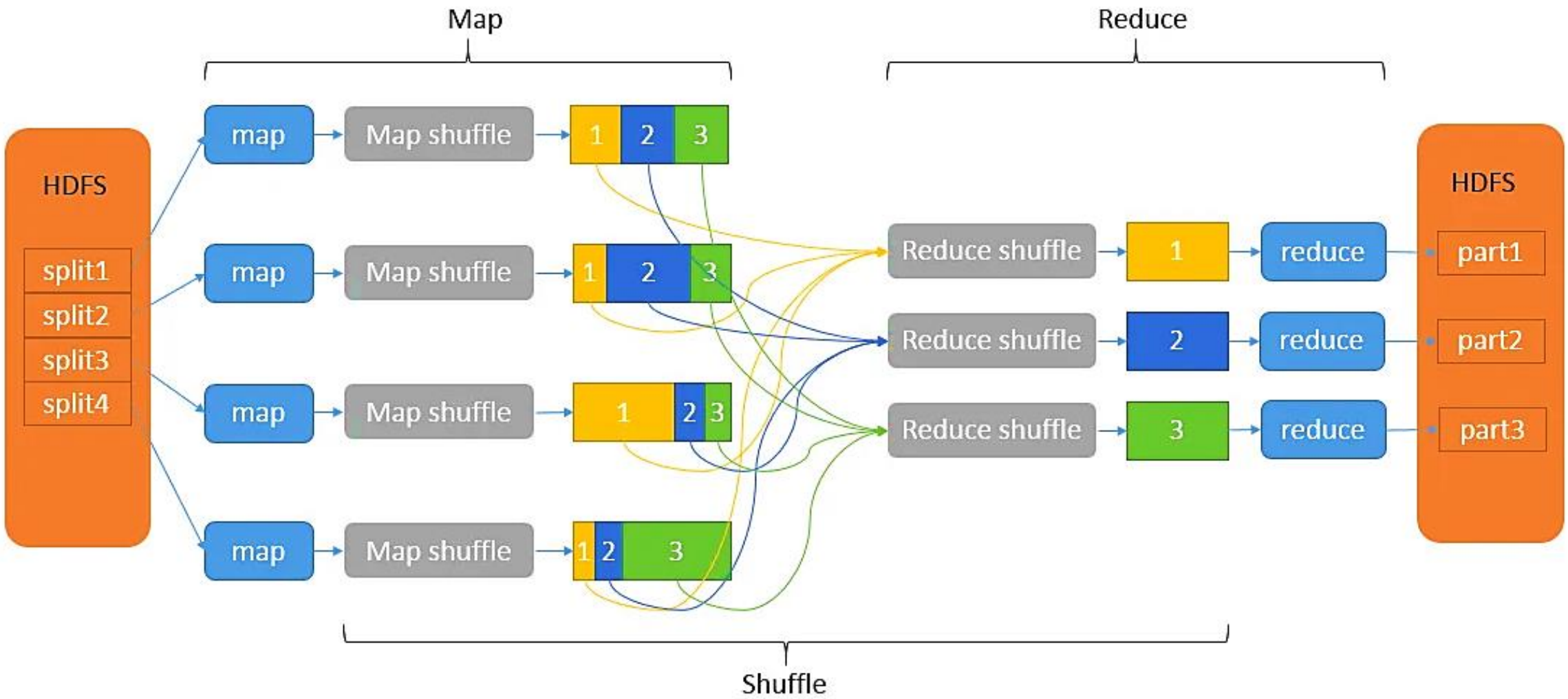
-OutputFormat: Le format de sortie spécifie la manière dont ces paires clé-valeur de sortie sont écrites dans les fichiers de sortie.



Bilan

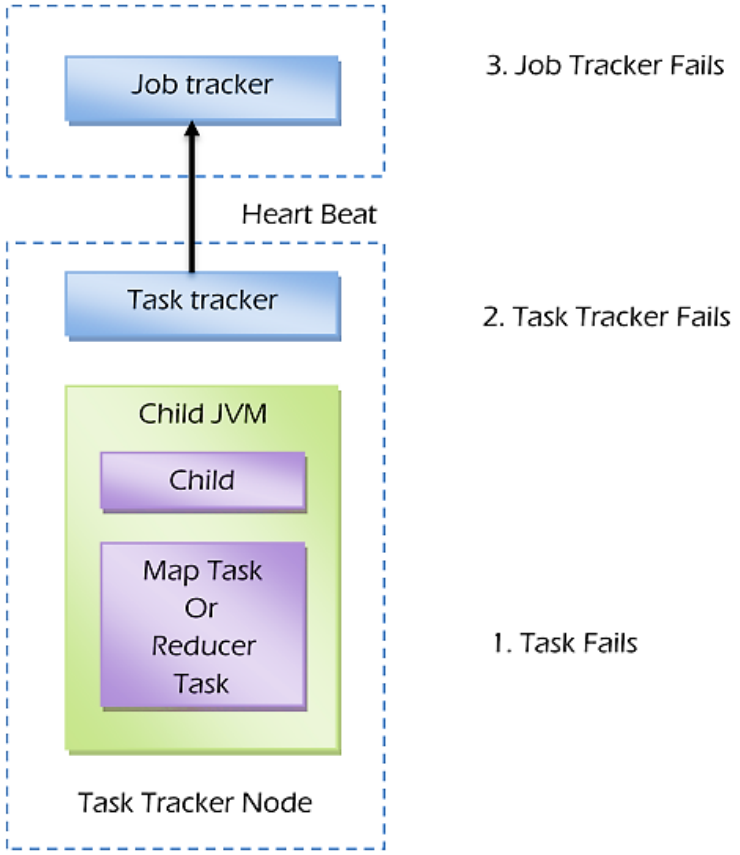

INPUT FILES
This is an Apple
Apple is red in color
MAPPER
This - 1
is - 1
an - 1
Apple - 1
Apple - 1
is - 1
red - 1
in - 1
color - 1
SHUFFLE
This - 1
is - 1
is - 1
an - 1
Apple - 1
Apple - 1
red - 1
in - 1
color - 1
REDUCER
This - 1
is - 2
an - 1
Apple - 2
red - 1
in - 1
color - 1
OUTPUT FILE
This - 1
is - 2
an - 1
Apple - 2
red - 1
in - 1
color - 1

Bilan


Map
Reduce
HDFS
split1
split2
split3
split4
map
map
map
map
Map shuffle
Map shuffle
Map shuffle
Map shuffle
1 2 3
1 2 3
1 2 3
1 2 3
Reduce shuffle
Reduce shuffle
Reduce shuffle
1
2
3
reduce
reduce
reduce
HDFS
part1
part2
part3
Shuffle

MapReduce: Tolérance aux pannes

- Echec de la tâche: redémarrer la tâche
- Echec de TaskTracker: En cas d'échec d'un TaskTracker pendant l'exécution de ses tâches Map, elles seront toutes attribuées aux autres TaskTrackers.
- Echec de JobTracker: Le master établit périodiquement des checkpoints de sa structure de données. Si le master est tué, une nouvelle copie est générée et reprends l'exécution au dernier checkpoint.



YARN: Yet Another Resource Negotiator

- YARN surmonte les limitations de MR1 grâce à son architecture de Ressource Manager/Application Master séparés: il est conçu pour évoluer jusqu'à 10 000 nœuds et 100 000 tâches au lieu de 40 000 tâches.
- MapReduce V1 a subi une refonte complète avec YARN, divisant les deux fonctionnalités principales de JobTracker (gestion des ressources et planification / surveillance des jobs) en parties distincts.

MapReduce 1	YARN
Jobtracker	Resource manager, application master, timeline server
Tasktracker	Node manager
Slot	Container

YARN: Exécution d'une Apps ds YARN