



Devoir surveillé – S1 – 2024/2025

Filière : Ing2_Info	Matière : Bases de données NOSQL et Big Data		Enseignante : Asma KERKENI
Date : 22 / 11 / 2024	Nbr de Crédits : 3	Coefficient : 3	Calculatrice : autorisée Documents autorisés : Non
Durée de l'épreuve : 1h	Régime d'évaluation : Mixte		Nombre de pages : 5
	EX (45%) + DS (22%) + TP (33%)		
Nom & Prénom :CORRECTION.....			Matricule :
Signature :	Code confidentiel :		Classe : N° Place :

X

Exercice 1 :

- Que signifie le terme "véracité" dans le contexte du Big Data ?

Fiabilité et précision des données dans le Big Data (véracité= authenticité - exactitude - fidélité – vérité...)

- Qu'est-ce qu'un Data Lake?

Stockage centralisé de données brutes, structurées, semi structurées et non structurées.

- Quelle cause majeure a entraîné l'explosion de données à l'origine de l'émergence du Big Data ?

L'essor d'Internet et des appareils connectés, démocratisation de la technologie, des réseaux sociaux....

- Comment le NameNode détecte-t-il les nœuds en panne dans Hadoop ?

Le NameNode surveille les signaux de cœur (heartbeat) envoyés par les DataNodes et détecte les nœuds en panne lorsque ces signaux ne sont pas reçus pendant une période déterminée.

- Quelle est la principale limitation qui freine la scalabilité de Hadoop dans sa version 1.

Le NameNode est le master de tous les jobs et peut devenir un goulot d'étranglement (c'est un SPOF).

- Quel est le rôle du Job History Server dans Hadoop 2 ?

Il stocke et fournit des informations sur l'historique des jobs MapReduce.

- Quel démon gère les ressources sur un noeud worker dans YARN ? NodeManager.

- Un fichier de 256 Go doit être stocké dans un cluster Hadoop avec une politique de réPLICATION de 2. Si chaque DataNode dispose de 50 Go de capacité libre, combien de DataNodes sont nécessaires (taille de bloc de 128 Mo) pour stocker le fichier ?

Il faut stocker: $256 \text{ Go} * 2 = 512 \text{ GO}$

DataNodes nécessaires : $512 \text{ Go} / 50 \text{ Go} = 10.24 \Rightarrow 11 \text{ DataNodes}$

- Le taux de parallélisme est défini comme le ratio entre les tâches simultanées et le total de tâches. Un job MapReduce contient 20 tâches Map, et le cluster dispose de 5 nœuds, chacun pouvant exécuter 3 tâches simultanément. Quel est le taux de parallélisme pour ce job ?

Taux de parallélisme= Nombre de tâches simultanées/ Nombre total de tâches = $15/20=75\%$

10. Pour traiter 512 Go de données dans un cluster Hadoop avec une taille de bloc de 128 Mo et 10 slots par nœud, combien de tâches Map seront générées, et combien de nœuds sont nécessaires pour un parallélisme optimal ?

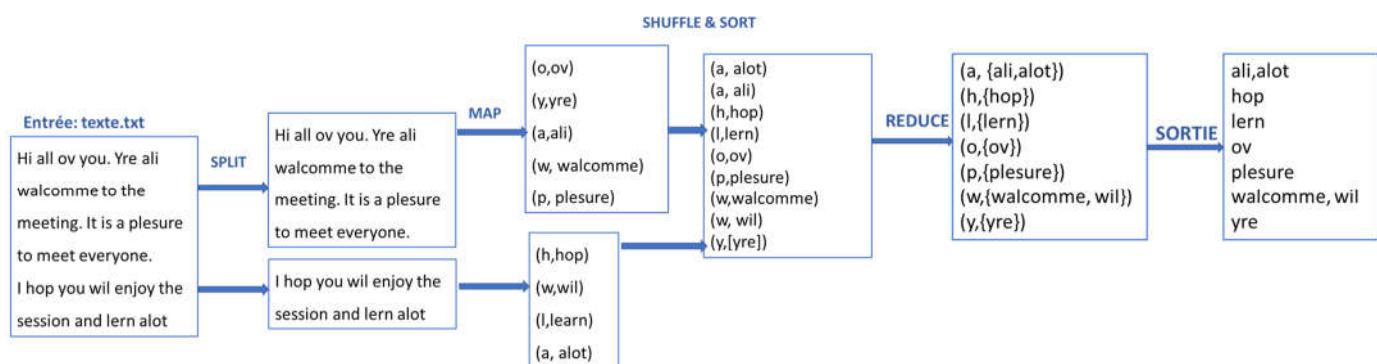
Nombre de tâches Map=Nombre de blocs=Taille totale des données/Taille du bloc=4096

Nombre de nœuds nécessaires pour un parallélisme optimal =

Nombre de tâches Map/Nombre de Slots par nœud = $4096/10=409.6 \Rightarrow 410$ (en SB ou 400 en SI)

Exercice 2 : (10 points)

1. Illustrer une solution utilisant le modèle de programmation *MapReduce* sur l'exemple.



Remarques :

- Même si l'exercice ne spécifie pas explicitement la clé de regroupement, le traitement Hadoop map reduce est basé sur les couples (clé, valeur) donc il faut retrouver la clé.
- Il est possible de modifier le résultat de sortie de reduce pour afficher ce qu'on veut (ici : les valeurs seulement)
- Lorsqu'on écrit un pseudocode de map reduce : on spécifie le traitement à faire pour chaque couple (**key, value**) pour le **map** et (**key, list_of_values**) pour le **reduce** : c'est différent (et plus simple) de l'implémentation en python streaming.

2. Ecrire les pseudocodes des fonctions *Map* et *Reduce* correspondantes à ce problème.

map(key, value): # input: key=offset, value=contenu de la ligne

création de dictionnaire (simplification pour cet exercice mais en production, on s'attend à un dictionnaire complet de la langue anglaise dans un fichier à part)

```
dictionary = {
    "a": ["a", "all", "and"],
    "e": ["enjoy", "everyone"],
    "h": ["hi"],
    "i": ["I", "is", "it"],
    "m": ["meet", "meeting"],
    "s": ["session"],
    "t": ["the", "to"],
    "w": ["will"],
    "y": ["you"]}
```

```
}

for word in value: # Parcourt chaque mot de la ligne
    word = clean(word) # clean : une fonction pour convertir en minuscules et supprimer
la ponctuation
    if word != "":
        first_letter = word[0] # Extraire la première lettre du mot
        # Vérifier si le mot est absent du dictionnaire
        if (word not in dictionary[first_letter]):
            Emit(first_letter, word) # Émettre (clé=première lettre, valeur=mot)
```

```
reduce(key, list_of_values): # key=first_letter, list of values=list des mots
    unique_words = set() # ensemble pour récupérer les mots sans répétition
    for word in list_of_values:
        unique_words.add(word) # ajouter le mot à l'ensemble
    sorted_words = sorted(unique_words) # trier les mots par ordre alphabétique
    Emit(key,sorted_words) # émettre la clé et la valeur puis modifier la sortie pour afficher
juste les clés
```