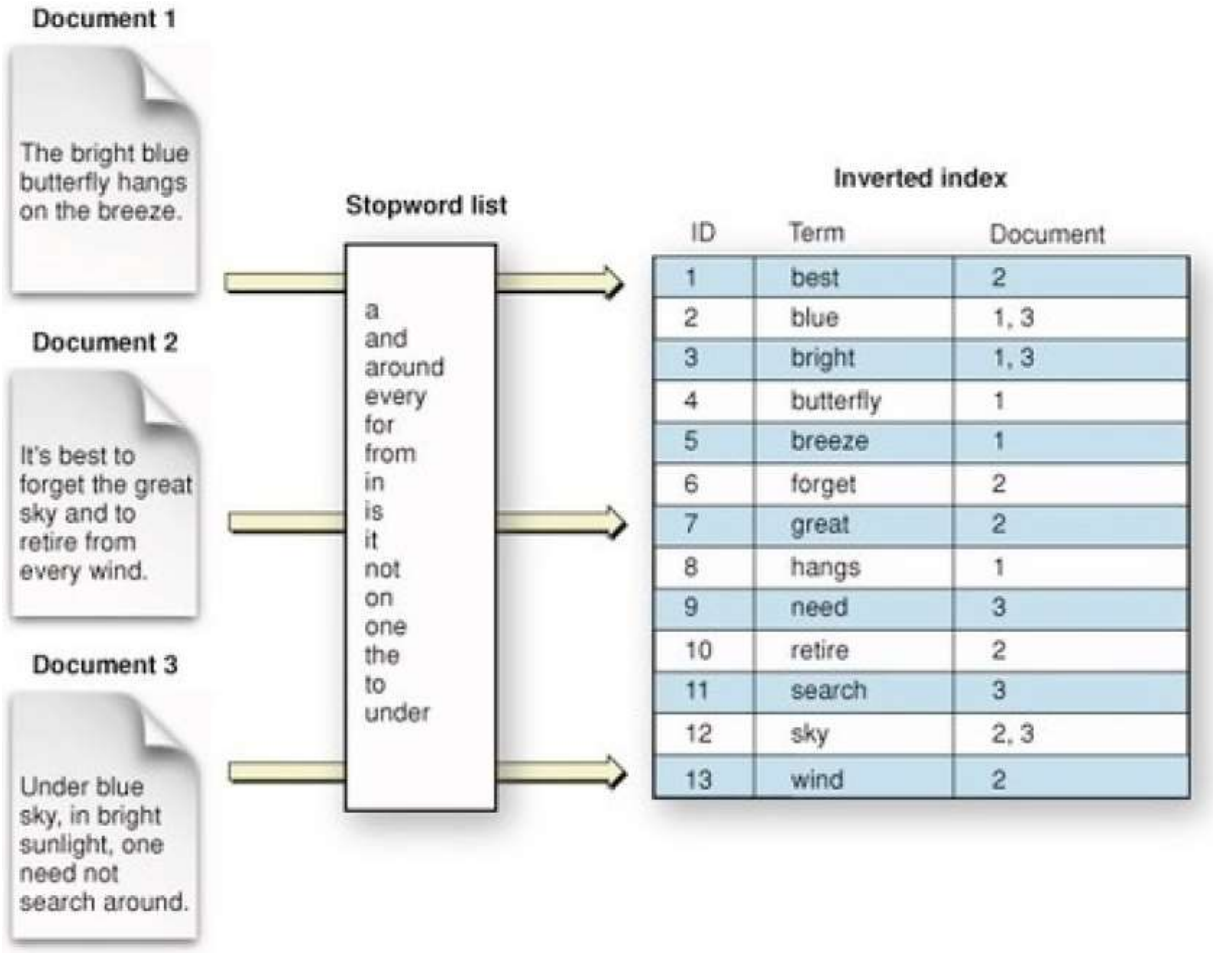
TD : Big Data - MapReduce

Exercice 1 : (Indexe inversé - Inverted Index)

Les index inversés sont des outils indispensables aujourd'hui. Les indexes permettent une recherche plus rapide. Avec un index inversé on se contente d'interroger cet index pour obtenir la liste des références sur les documents contenant les termes recherchés. Par exemple, les indexes à la fin d'un livre (cf figure 1), ou les index utilisés par google pour référencer les pages web pour un mot donné.

Exemple :



Question :

Proposez une solution algorithmique MapReduce permettant de créer à partir d'un corpus de documents un index inversé.

Exercice 2 : (Calculs de moyenne pondérée)

On considère des rafales de mesures regroupés dans des Records, stockés dans un fichier Hadoop (dans le système de fichiers HDFS), et représentant les sorties de différents capteurs :

{ moteurID : RefDuMoteur,

capteurID : RefDuCapteur,

rafale : [v0,…, v9],

unité : UnitéDeLaMesure,

fiabilitéCapteur : PoidsDeLaMesure }

On peut ainsi analyser de multiples mesures concernant l'ensemble des moteurs. Afin d'obtenir un système plus fiable, plusieurs capteurs sont installés sur chaque moteur, qui mesurent la même grandeur, mais qui n'ont pas tous la même fiabilité.

Question 1 : Calcul de la valeur moyenne d'une série de rafales

- Proposez une solution algorithmique Map-Reduce permettant de calculer le valeur moyenne de toute une série de rafales concernant un même moteur. Chaque mesure participera à la moyenne en étant pondérée par le coefficient de fiabilité de son capteur. Concevez une solution simple, constituée seulement de Mappers et de Reducers.

Question 2 : Solution optimisée en Hadoop

Utilisez toute la puissance d'expression d'Hadoop pour implanter une solution optimisée, qui notamment ne fasse pas tous les calculs dans les Reducers, et surtout qui réduise le trafic dans la phase « Shuffle »

- Définissez les paires clé-valeurs de votre solution (si besoin elles peuvent être différentes de celles de la solution précédente)
- Indiquez le pseudo-code de vos différentes tâches

Question 3 : Elimination de mesures aberrantes et détection des capteurs suspects

En fait, toutes les mesures ne sont pas fiables. Il arrive qu'un capteur soit perturbé et produise une valeur aberrante. Nous considérerons qu'une valeur de mesure inférieure à -100 ou supérieure à +100 est une valeur aberrante.

- Concevez une solution Map-Reduce qui, pour chaque moteur, élimine les valeurs aberrantes du calcul de la moyenne **et** qui construise de la liste des capteurs suspects ayant généré au moins une valeur aberrante.

Exercice 3 : (Graphe Social)

On suppose qu'on administre un réseau social comportant des millions d'utilisateurs. Pour chaque utilisateur et à partir d'une base de données, on peut disposer de la liste des amis de cet utilisateur sur le réseau (via une requête SQL). Quand un utilisateur va sur la page d'un autre utilisateur, on souhaite indiquer le message suivant « Vous avez N amis en commun ». On ne peut pas se permettre d'effectuer une série de requêtes SQL à chaque fois que la page est accédée (traitement lent).

On va donc développer des programmes Map-Reduce pour effectuer cette opération et exécuter le traitement toutes les nuits sur notre base de données, en stockant le résultat dans une nouvelle table.

Nous supposons que nos données sont sous la forme d'un graphe d'adjacence stocké dans un fichier où chaque ligne renseigne les amis d'une personne. Une ligne commence ainsi par le numéro d'un utilisateur suivi par les numéros de ses amis.

- Proposez un algorithme Map-Reduce qui permet de calculer pour chaque couple de personnes le nombre et la liste de leur amis communs

Exercice 4 : (Recommendation)

Une entreprise de vente de produits en ligne possède un fichier sauvegardant pour chaque client les numéros de ses commandes effectuées et la liste des produits dans chaque commande. Plus précisément, ce fichier se présente sous le format d'une succession de lignes sous la forme : id_client id_commande id_produit_1 id_produit_2 ... id_produit_N

Une ligne du fichier contient donc l'identifiant d'un client, l'identifiant d'une commande qu'il a effectué, les identifiants des produits de la commande. Bien entendu, un client peut avoir plusieurs commandes enregistrées, et un produit peut être enregistré dans plusieurs commandes.

Pour améliorer son système de recommandation, l'entreprise souhaite connaitre les produits qui se vendent le mieux ensemble. Pour cela, nous utiliserons le paradigme MapReduce avec Hadoop.

- Nous souhaitons calculer pour chaque pair de produits (idp1,idp2), apparaissant au moins une fois dans une même commande, le nombre de clients ayant achetés au moins une fois le produit idp1 et le produit idp2 simultanément dans une même commande.