Ce TP a pour objectif de vous familiariser avec les plateformes de Big Data afin de comprendre les principes fondamentaux du fonctionnement de ces systèmes reposant sur une parallélisation massive.

En réalisant ce TP, vous serez mieux préparés à tirer pleinement parti des séminaires industriels sur le Big Data, dans lesquels ces concepts sont évoqués.

Enfin, les exercices proposés constituent également une bonne préparation pour l'examen ;-)

Travail demandé et format du rendu

1. Le TP est à faire en binômes.
2. Il s'agit d'écrire des programmes avec la bibliothèque Hadoop-Map/Reduce répondant aux questions.
3. Il est demandé de
    - rédiger un document (3 pages max, annexes exclus) expliquant comment vous avez répondu à chaque question du TP
    - créer un dépôt git avec vos solutions et de l'indiquer (a) dans votre rendu et (b) dans le document ici (https://umontpellierfr-my.sharepoint.com/:x:/g/personal/federico_ulliana_umontpellier_fr/EUPXJoYsjWNPiVs_aufMgXYBR6qykXgO3-sKawZOEwDM8w?e=rWnVRb) ; si le lien n'est pas fourni ou le projet est inacessible (prévoir un test) le TP ne pourra pas être évalué. Dans le document, est important de mettre en évidence juste les points les plus importants vous ayant permis de répondre à la question. Le code Java n'est pas à déposer sur Moodle, car disponible dans votre git.

Il est important de lire tout le document.

Ressources

- [MR1] MapReduce: Simplified Data Processing on Large Clusters - Jeffrey Dean and Sanjay Ghemawat
- [MR2] Apache Hadoop http://hadoop.apache.org/ (http://hadoop.apache.org/)
- [MR3] Hadoop: the definitive guide (http://grut-computing.com/HadoopBook.pdf (http://grut-computing.com/HadoopBook.pdf))

Avant commencer

La programmation en Map-Reduce utilise le langage Java - avec ses avantages et inconvénients. Il est ainsi indispensable d'effectuer ce travail en binômes, de rester bien concentrés pour bien comprendre la cause des bogues (souvent ce seront des problèmes de typage ou nommage des ressources).

Installation

Disclaimer : : Le TP a été testé et fonctionne pour les configurations suivantes : :

1. Ordinateurs de la faculté, Ubuntu (Eclipse, IntelliJ)
2. Ordinateurs personnels, Linux ou OSX (Eclipse, IntelliJ)

Windows n'est pas supporté (voir ici (https://issues.apache.org/jira/browse/HADOOP-7682)). **Pourquoi ?** Car Hadoop est un système **distribué** complèxe qui, dans ce TP, sera exécuté de façon centralisée (sur votre compte informatique) pour des raisons pratiques. Cela pose plusieurs défis techniques sous Windows qui ne sont pas adressés ici.

Pour contourner les différences entre systèmes d'exploitation et environnements de développement (IDE), deux versions du TP sont mises à disposition. Essayez d'abord d'importer et de tester le projet avec la version v1.2. Si tout fonctionne correctement, vous pouvez passer à la section suivante.

Dans votre IDE, il est probable que vous deviez :

- configurer Kerberos (sécurité)
- configurer Java (version)

Troubleshooting :

- Problème avec `Kerberos` : ajouter variable d'environnement `HADOOP_USER_NAME`, voir ici (https://gitlab.etu.umontpellier.fr/p00000013857/edbd/-/blob/main/TP4_Hadoop-MapReduce/doc/install.md#probl%C3%A8me-sur-les-nouveaux-postes-informatique)
- Problème `security manager not active' avec IntelliJ. L'IDE IntelliJ pourrait utiliser une JVM différente de celle indiquée dans le` pom.xml`, ce qui produit une erreur concernant le Java security manager. Dans ce cas, indiquer explicitement l'utilisation de Java 11 pour votre projet. Dans le menu File > Project Structure > Project > Project SDK. forcer l'utilisation de Java 11, voir ici (https://gitlab.etu.umontpellier.fr/p00000013857/edbd/-/blob/main/TP4_Hadoop-MapReduce/doc/install.md#probl%C3%A8mes-de-securit%C3%A9-forcer-lutilisation-de-java-11)
- Problème `file path does not exist' : changer les chemins des fichiers correspondant aux variables` INPUT_PATH`et`OUTPUT_PATH`dans`WordCount.java`

FAQ

Le log dans la sortie standard est illisible. Je ne comprends pas ce que fait mon programme Map-Reduce !
Réponse : Le programme lit les fichiers contenus dans le répertoire `input-wordCount/`, puis effectue un comptage des mots qui est sauvegardé dans le repertoire `output` (vérifiez-le !).

Qu'indiquent les statistiques d'éxécution du programme Map-Reduce ?

Voir les détails ici (doc/stats.md)

Où sont les résultats de mon programme Map-Reduce ?
Voir le répertoire `output/wordCount-xxxx`.

Y a-t-il de la parallélisation à l'état actuel ?
Non, on utilise Hadoop en mode Standalone pour ce TP (les plus courageux peuvent essayer la version "pseudo-distributed").

Qu'est ce qu'on fait maintenant ?
Modifiez les programmes fournis pour implémenter les traitements ci-dessous. Comprenez d'abord le code.

Important : Comment puis-je déboguer mon programme ?

Vous disposez d'un logger via la variable de classe `LOG`. Ces messages sont aussi enregistrés dans `TP_HMIN122M-hadoop/out.log`. Utilisez également le debugger d'Eclipse (Run/Debug ou `F11`).

Exercices de préparation - Partie 1

Exercice 0 - WordCount

Consulter le document fourni dans le dossier `input-wordCount` pour avoir un aperçu du contenu. Ensuite, tester le programme WordCount.

Exercice 1 - WordCount + Filter

Modifier la fonction reduce du programme WordCount.java.

1. Afficher uniquement les mots ayant un nombre d'occurrences supérieur ou égal à 10
2. Supprimer la ponctuation, mettre le texte en minuscule, et exclure tous les mots de taille inférieure ou égale à 4 lettres de l'analyse
3. Regardez le résultat : quels est le mot le plus fréquent ?

Exercice 2 - Group-By

Implémenter un opérateur de regroupement sur l'attribut `Customer-ID` dans GroupBy.java.
Les données sont dans `input-groupBy` et doivent calculer le total des profits (`Profit`) par client.

Exercice 3 - Group-By

Modifier le programme précédent :

1. Calculer les ventes par `Date` et `State`.
2. Calculer les ventes par `Date` et `Category`.
3. Calculer par commande :
    - Le nombre de produits distincts achetés.
    - Le nombre total d'exemplaires.

Exercice 4 - Join

Créer une classe Join.java pour joindre les informations des clients et commandes dans `input-join`.
Restituer les couples `(CUSTOMERS.name, ORDERS.comment)`.

Note : Copier les valeurs de l'itérateur dans un tableau temporaire et utiliser deux boucles imbriquées pour effectuer la jointure.

Exercices - Partie 2

À l'aide de map/reduce, implementer trois (3) requêtes analytiques proposées pour le premier datamart (aspect principal) de votre projet et deux (2) requêtes analytiques proposées pour le deuxième datamart (aspect secondaire).

Vous pouvez rapidement extraire vos données de votre instance Oracle avec les commandes suivantes que vous pouvez adapter pour vos tables.

```
-- ouvrir la connexion
SET MARKUP CSV ON;

-- répeter pour chaque table à exporter
SPOOL change_this_table_name.csv;
SELECT * FROM change_this_table_name;

-- dernière commande avant de fermer la connexion
SPOOL OFF;
```

Un lien vers des exercices facultatifs, mais utiles pour la préparation à l'examen

Vous trouverez ici (more/advanced_questions.md) une liste de question que vous pouvez utiliser pour travailler plus en profondeur la programmation en map/reduce.

Windows n'est pas supporté.

Vous pouvez éventuellement essayer d'exécuter une machine virtuelle Linux depuis votre système Windows. Une altre possibilité est d'installer Hadoop sous Windows. Ces deux solutions peuvent prendre du temps, et notre conseil est d'utiliser les comptes informatiques de la FdS.