Projet Données Réparties

Mohammed NAHI(L4)/Achraf KHAIROUN(L3)

Département Sciences du Numérique - Deuxiéme année
2023-2024

1 Manuel d'utilisation

1. Configuration des Machines Distantes (HDFS Servers)

   Il faut spécifier les machines distantes qui agiront en tant que serveurs HDFS. il faut remplir le fichier `listeMachines.txt` situé dans le dossier `config` en ajoutant les noms ou adresses IP des machines distantes, un par ligne. :

   Il faut s'assurer que chaque ligne représente le nom ou l'adresse IP d'une machine distante qui agira en tant que serveur HDFS. Ces machines seront utilisées pour la distribution des fragments de fichiers et l'exécution des tâches du projet.

2. Exécution du Serveur HDFS

   Il faut ensuite exécuter le serveur HDFS sur chaque machine distante spécifiée dans le fichier `listeMachines.txt`. Le serveur HDFS permettra la gestion des commandes d'écriture, de suppression et de lecture sur les fichiers.Cela est fait en exécuteant le serveur HDFS en spécifiant le port sur lequel il écoutera les connexions. Les ports doivent commencer par 6001 et augmenter pour chaque machine :

   Si vous avez plusieurs machines, attribuez un port unique à chaque machine en augmentant le numéro du port (par exemple, la première machine utilise 6001, la deuxième utilise 6002, etc.). Assurez-vous d'utiliser des ports différents pour chaque instance du serveur HDFS sur des machines différentes. Répétez ces étapes sur chaque machine distante listée dans le fichier `listeMachines`.

   Assurez-vous que le serveur HDFS est correctement démarré sur toutes les machines avant de passer à l'étape suivante. Vous devriez voir des messages de confirmation tels que "Serveur démarré sur le port : (numero de port)" sur chaque machine.

3. Exécution du Worker sur Chaque Serveur

   Après avoir configuré les serveurs HDFS, il est maintenant temps de lancer le Worker sur chaque machine. Le Worker est responsable de l'exécution des tâches de mapping sur les fragments de données.

   (a) Compilez le fichier `WorkerImpl.java` sur chaque machine distante :

   ```
   javac WorkerImpl.java
   ```

   (b) Exécutez le Worker sur chaque machine :

   ```
   java WorkerImpl
   ```

   (c) Répétez ces étapes sur chaque machine distante listée dans le fichier `listeMachines`.

   Le Worker doit etre correctement démarré sur toutes les machines avant de passer à l'étape suivante.

4. Exécution du Client HDFS

   Le client HDFS est responsable de l'interaction avec le système HDFS pour écrire, lire ou supprimer des fichiers. Voici comment exécuter le client HDFS pour écrire un fichier texte.

   (a) Compilez et executer le fichier `HdfsClient.java` sur la machine cliente :

   ```
   java HdfsClient write line nom_fichier_a_traiter.txt
   ```

   Cette commande divise le fichier en fragments et les envoie à chaque serveur spécifié dans le fichier `config/listeMachines.txt`.

Ces étapes supposent que le fichier texte que vous écrivez est situé dans le dossier `data` du client. Vous pouvez également utiliser d'autres formats tels que `kv` en remplaçant `line` par `kv` dans la commande d'écriture.

Pour lire ou supprimer des fichiers, vous pouvez utiliser les commandes suivantes :

```
java HdfsClient read  nom_fichier_a_traiter.txt
java HdfsClient delete  nom_fichier_a_traiter.txt
```

La commande `read` lit tous les fichiers générés après l'exécution des `runMap` sur chaque serveur et les regroupe dans un fichier appelé `ReasultatRead.txt` dans le dossier `data` du client.

5. Exécution du JobLauncher

   Le JobLauncher est responsable de la coordination du processus de MapReduce sur les serveurs distants, l'exécution de la lecture, de la suppression et de la réduction finale des résultats.

   (a) Compilez et executer le fichier `JobLauncher.java` sur la machine cliente :

   ```
   java JobLauncher fichier_a_traiter format
   ```

   Remplacez `fichier_a_traiter` par le nom du fichier que vous souhaitez traiter et `format` par le format approprié (0 pour `line`, 1 pour `kv`).

   (b) Le JobLauncher exécutera la phase de Map sur chaque serveur distant. Chaque fragment du fichier sera traité par un Worker distinct.

   (c) Après l'exécution des tâches Map, le JobLauncher effectuera les opérations de lecture (Read) et de suppression (Delete) sur les fichiers générés par les Workers.

   (d) Enfin, le JobLauncher effectuera la phase de Reduce en regroupant les résultats lus dans un fichier final.

2 Limitations et Améliorations

(a) Le système actuel n'effectue pas une gestion approfondie des exceptions en cas d'arguments non cohérents fournis par l'utilisateur lors de l'exécution du JobLauncher. Des mécanismes supplémentaires peuvent être ajoutés pour vérifier la validité des arguments et informer l'utilisateur de manière claire en cas d'erreur.

(b) Actuellement, le JobLauncher n'attend pas la terminaison de la phase de Map sur chaque serveur avant de passer à l'étape suivante. Cela peut conduire à des résultats imprévisibles si les opérations de lecture, suppression et réduction sont déclenchées avant la fin de la phase de Map. Une amélioration consisterait à mettre en œuvre un mécanisme d'attente, garantissant que toutes les tâches Map sont terminées avant de passer à la suite.

(c) Le système n'a pas été testé de manière approfondie sur des fichiers de grande taille. Des tests supplémentaires sur des données volumineuses sont nécessaires pour évaluer les performances et la scalabilité du système. Cela pourrait impliquer l'utilisation de fichiers de plusieurs gigaoctets pour s'assurer que le système peut gérer efficacement des charges de travail importantes.

(d) La classe `NetworkReaderWriter`, bien que présente dans les interfaces, n'a pas été utilisée dans notre implémentation. Cela est dû à une compréhension incomplète ou insuffisante de son utilité dans le contexte du projet.