RAPPORT

Projet Données Réparties
Evaluation Hidoop v0

Groupe L1

ALAUZY Estelle
DAVIN Marie
RONGIER Chloé

Sommaire

Introduction

L'objectif de ce projet est de gérer des traitements d'un grand volume de données. Pour ce faire, on souhaite découper les données en fragments stockés sur des machines de traitement. Le traitement des données peut ainsi se faire en parallèle sur plusieurs machines. Les résultats du traitement sont renvoyés et concaténés pour produire un résultat final.

Notre trinôme a implémenté une première version du service HiDFS (Hidoop Distributed File System). Nous évaluerons donc dans ce rapport la version du service Hidoop réalisé par l'autre binôme (Adrien LAPORTE, Chervin AMIRKAVEH).

Partie technique

I - Tests

La seule chose testable à ce point est le bon lancement du démon.

Nous avons ainsi testé la classe DaemonImpl en lui donnant pour argument une chaîne de caractères (nom souhaité du daemon), le daemon se lance bien.

Nous avons essayé de compiler MapReduce en sachant qu'il était impossible de ne pas avoir d'erreur en considérant qu'il manque toute la partie Hdfs au code (nécessaire à son exécution). Nous avons cependant pu observer les erreurs. Certaines n'étaient pas prévues. Il semblerait notamment que CallBackImpl ait besoin d'être Serializable.

II - Proposition de corrections

- Les différents démons utilisent tous le même URL. Or, il serait souhaitable que chacun ait son propre URL. Pour cela, le nom du démon pourrait remplacer "/Daemon" dans l'URL.
- Nous avons aussi remarqué que dans le fichier LineFormat, dans la méthode read, en ligne 19, "getIndex" est utilisé et la méthode employée pour que le programme l'accepte comme String nous paraît un peu brouillon. Il aurait été plus propre d'utiliser un "longtoString" plus correct.
- Il faudrait s'intéresser à la nécessité de rendre CallBackImpl Serializable.

III - Qualité du code

La plupart des fonctions importantes ont été correctement commentées. Certains endroits manquent cependant d'explications : la classe CallBackImpl et le main de DaemonImpl.

Les différentes classes n'ont pas été documentées.

L'optimisation du code a été assez bien gérée, notamment pour la gestion des formats, une classe abstraite FormatImpl a été créée implémentant l'interface Format. Cela leur a permis d'écrire une seule fois les méthodes open, close, getIndex, getFname et setFname. Cependant, ils ont initialisé des arguments inutiles (FileR et buffW) qui ne sont pas utilisés.

Un fichier "Project.java" a été initialisé dans le package "config". Ce fichier regroupe les informations à propos des ports et des noms des serveurs. Le fichier ne sert pas encore beaucoup, mais nous trouvons l'idée bonne pour les prochaines versions.

Synthèse

I - Correction

Nous n'avons pas pu tester la correction sans la combinaison entre les 2 parties. Nous avons testé le lancement du démon, qui paraît se lancer correctement. Nous avons aussi essayé de lancer la classe MapReduce et une erreur semble indiquer que CallBackImpl devrait être Serializable.

II - Complétude

- Les map sont lancés de façon successive et non parallèlement.
- Il manque l'interpréteur de commande.

III - Pertinence

Le travail exécuté correspond à ce qui est demandé pour un premier rendu. Le principe semble compris et l'architecture globale du programme est cohérente et réfléchie.

IV - Cohérence

Une classe abstraite a été utilisée dans Format permettant d'éviter la redondance du code.

L'architecture des différents programmes semble cohérente et les programmes sont concis.

Nous avons cependant noté une erreur dans l'expression de l'URL qui ne diffère pas pour chaque démon. Le nom choisi devrait y figurer afin d'avoir un nom distinctif pour chaque démon lancé.

V - Pistes d'amélioration

AMÉLIORATION	**RAISON**
Donner un URL propre à chaque démon	Pouvoir appeler chaque démon
Lancer des threads map en parallèle	Optimiser le temps de calcul
Retirer les arguments inutiles de FormatImpl	Pertinence du code
Rendre Serializable CallBackImpl	Pouvoir échanger les CallBack en RMI