Réponses à l'évaluation croisée

Armelle Jezequel
Loïs Roth
Paul Lacaille
Yannis Benabbi
Alexandre Chatain

15 janvier 2020

1 Hidoop

1.1 Réponses aux propositions de tests

Faire des tests unitaires s'est avéré un peu compliqué, cependant nous avons pu tester l'équivalent en utilisant un scénario de test proche du scénario 2 proposé. Le détails de nos tests sont dans le rapport final.
Ces tests nous ont part ailleurs permis de vérifier qu'à l'exécution le déroulement du map/reduce se faisait bien de la façon supposée dans le paragraphe Pertinence.

1.2 Réponses aux pistes d'améliorations

Nous n'avons pas implémenté de système de barrière, car la façon dont nous procédons, même si elle n'est peut-être pas optimale, fonctionne correctement, et nous avons préféré nous concentrer sur la résolution d'autres problèmes (faire fonctionner les daemons sur des machines distantes par exemple).
Concernant la gestions des daemons, la propositions selon laquelle les daemons renseignent eux-mêmes le nom de leur machine est intéressante, mais conceptuellement problématique, cela pour deux raisons :

- Tout le principe de rebind/lookup est d'établir un canal de communication. Or, si les Daemons peuvent communiquer le nom de leur machine au Job, il existe déjà un canal de communication, donc l'opération devient inutile.
- La raison pour laquelle cela pourrait fonctionner est que les Daemons et le Job sont lancé depuis la même session et ont accès au fichier Projet.java. Toutefois, ce projet est destiné à nous faire comprendre ce qu'est une application client serveur, dans laquelle les appels à des services distants se font à travers un échange de requêtes plutôt que par des fichiers partagé. C'est pourquoi nous pensons que ce serait contre l'esprit du projet.

2 HDFS

2.1 Réponses aux propositions de tests

Le code de la partie HDFS est maintenant fonctionnel mais le fait d'implémenter des tests unitaires pour les méthodes reste compliqué car elles font pour la plupart appel aux InputStream et OutputStream des socket. Nous avons donc tester les trois fonctionnalités du programme séparément comme c'était proposé.
Pour que les tests soient pertinents, il faut toujours envoyer un fichier sue le serveur, mais les daemons n'ont plus de problèmes même si on demande à lire un fichier sans l'avoir envoyer auparavant. Après avoir donc envoyer un fichier, on peut voir que de nouveaux fichiers sont créés

par les daemons. On peut ensuite supprimer le fichier d'origine et lire le fichier pour qu'il soit recréer à l'identique. Enfin, la suppression supprime les fichiers créés par les daemons. Les trois fonctionnalités fonctionnent donc comme on le veut.

2.2 Réponses aux pistes d'améliorations

Les noms et les ports des daemons sont maintenant stockés dans un fichier de configuration qui est partagé avec la partie Hidoop afin que les deux parties restent cohérentes. Seul l'exécution des daemons demande un argument, l'identifiant du daemon.
Nous avons de plus ajouter une classe Fragmenteur qui contient la méthode de fragmentation du contenu du fichier envoyer sur le serveur. Nous avons ajouté cette classe dans le but de généraliser le programme et permettre de lui ajouter plus facilement des extensions. Il reste cependant encore quelques classes à ajouter afin d'abstraire certains comportement du programme.