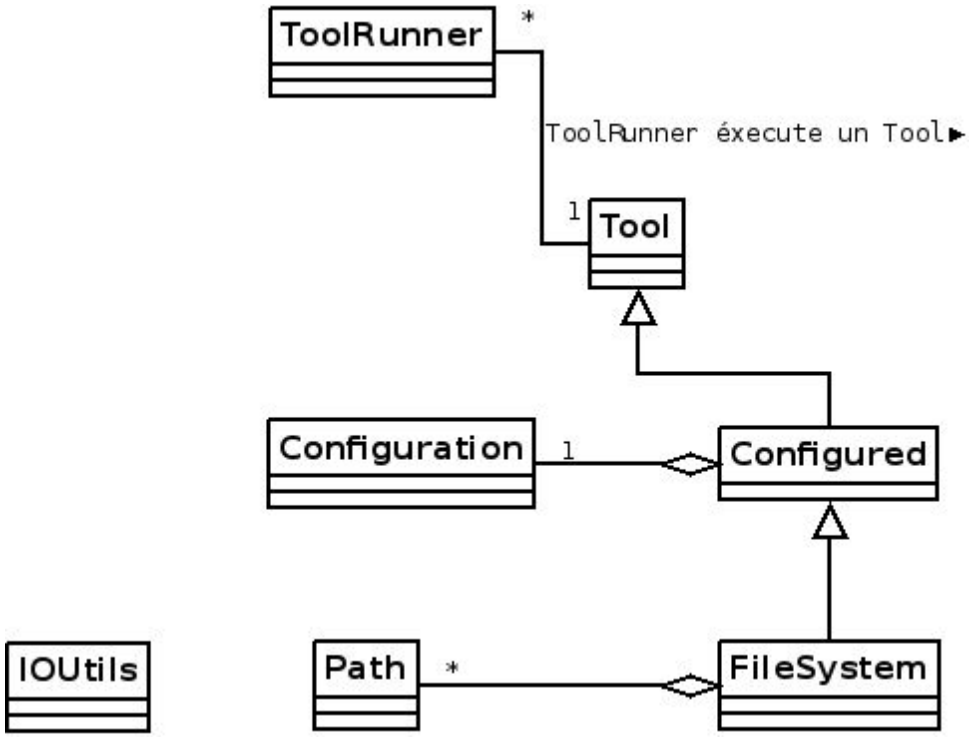
Bérénice Faltrept
Kevin Baptista

Rapport TD2 - Partie 2 : API

Préalablement, nous souhaitons vous informer que nous avons joint le jar exécutable à ce mail car le cluster utilisé en TP est actuellement clos.

Nous avons réuni les exercices en un seul exécutable qui vous affiche un usage si vous ne lui indiquez aucun argument : hadoop jar MyApplication.jar

Exercice 5 : API Hdf



IOUtils semble parfaitement déconnecté sur notre graphe car nous n'avons pu le relier directement aux classes demandées ici. C'est un utilitaire dont tout le monde peut-être amené à se servir. Nous n'avons pas vu d'indices pour déterminer qui.

Exercice 6 : Premier Programme (CopyFromLocal)

Nous avons utilisés le code que vous nous avez proposés dans le cours et l'avons intégré dans notre application.
Nous avons lancés un nouveau cluster et nous nous sommes aperçus que les droits d'accès au système de fichiers du cluster nécessitaient de s'authentifier. Ne sachant pas les droits du cluster que vous allez utiliser pour les tests, nous avons ajoutés cette information en argument.

Nous pensons avoir compris le fonctionnement et les relations entre les objets de ce programme : On passe en paramètre ce dont on a besoin (URI, chemin du fichier à copier …), on crée les flux d'entrée et de sortie et on lance la copie à l'aide de la classe IOUtils. On termine en fermant les flux.

Exercice 7 : Concaténation de fichiers (MergeFromLocal)

Nous avons là aussi utilisés le code que vous nous avez proposés à la fin du cours de cette semaine.
Le principe est le même que celui de l'exercice 6. La seule différence est que l'on copie plusieurs fichiers les uns à la suite des autres. On a besoin de plusieurs chemins d'entrée, on crée donc une boucle pour parcourir les arguments et lancer la copie de chaque fichier.

comparaison d'efficacité avec le script

> time ./script.sh	> time hadoop jar MyApplication.jar MergeFromLocal ...
real 0m25.998s user 0m4.796s sys 0m1.196s	real 0m10.016s user 0m5.116s sys 0m0.572s

C'est un résultat que l'on ne s'explique pas pour le moment. Notre script génère le fichier localement puis fais une seule requête au namenode alors que l'application va faire une demande par fichier à ajouter. On s'attendait à ce que le script ait de meilleurs résultats que l'application.

Voici notre script

```
find . -type f -exec cat {} >> source 2> /dev/null \;
hdfs dfs -copyFromLocal source  hdfs://lentix:9000/source
rm source
```

Exercice 8 : Générateur de mots aléatoires (Random Words)

Pour créer les mots, nous avons répertorié un ensemble des syllabes japonaises (choix arbitraire pour faire des mots jolis même s'ils ne veulent rien dire). Nous choisissons ensuite un nombre aléatoire allant de 0 au nombre de syllabes répertoriées de manière pseudo aléatoire qui détermine une de ces syllabes. On boucle dessus autant de fois que demandé pour créer le mot de la longueur voulue.
Il est possible de créer plusieurs mots de différentes longueur en passant plusieurs entier en paramètre.

On a choisi d'écrire les divers mots dans un String qui est ensuite envoyé via une seule demande au namenode afin de limiter les problèmes de latences induits par de multiples demandes.