# RAPPORT FINAL : Partie Hidoop

Luc Bourguignon et Sherwin Sabras

## 1/ Les modifications par rapport au premier rendu :

Notre rendu Hidoop du premier semestre comportait des erreurs que nous avons eu du mal à réparer. Mais nous avons finalement réussi à rendre notre application complètement fonctionnelle. Cette dernière version permet donc, comme attendu, de réaliser un comptage de mot réparti sur un nombre de machines (cluster) défini préalablement, puis de récupérer le résultat sous forme d'un seul fichier texte.

## 2/ Déploiement de l'application :

Grâce à des scripts Bash, le lancement de notre application est simplifié. Ces scripts permettent une automatisation des lancements :

- *test_perf.sh* : ce script lance le déploiement de l'application HDFS (démons sur les machines distantes, client en local) et Hidoop (démons et client) en prenant en entrée les valeurs du fichier de configuration : noms des machines, ports utilisés et nombre total de fragments du fichier. On prend soin d'arrêter toutes les instances précédentes potentiellement encore présentes. Ce programme mesure également le temps total d'exécution (mm:ss) de l'ensemble HDFS + Hidoop (envoi des fragments et réduction) ;

- *reinisia.sh* : Ce script permet de supprimer les dossiers contenant les fragments et les données produites par HDFS et Hidoop (afin de pouvoir relancer test_perf.sh "proprement") ;

- *deploiement_hdfs* : lancement des démons HDFS et du client avec les paramètres demandés ;

- *deploiement_hidoop.sh* : idem avec l'application Hidoop.

## 3/ Outils supplémentaires :

Afin de simplifier les tests, nous avons ajouté deux scripts (codés en python) :

- *generer_texte.py* : le premier permet de générer des fichiers .txt de grande taille (que l'on peut choisir grâce à une variable N directement proportionnelle à la taille du fichier) ainsi qu'un fichier resultat_attendu.txt qui contient les mots et le nombre de fois où ils sont présents ;

- *comparaison_hidoop.py* : le second facilite la vérification du résultat donné par l'application : il compare le nombre d'itération de mots trouvés par notre application (*resultat_obtenu.txt*) avec resultat_attendu.txt (dont le contenu est décrit plus haut). En cas d'erreur le script précise le mot qui pose problème et la différence constatée entre la valeur obtenue et la valeur attendue.

Nous avons également pu avancer sur l'algorithme de Monte-Carlo (compréhension principalement), mais n'avons pas pu avancer plus faute de tests de performances. Nous avons pu aider à la correction d'erreurs et au debug sur HDFS ainsi que sur les scripts de déploiement de l'application.