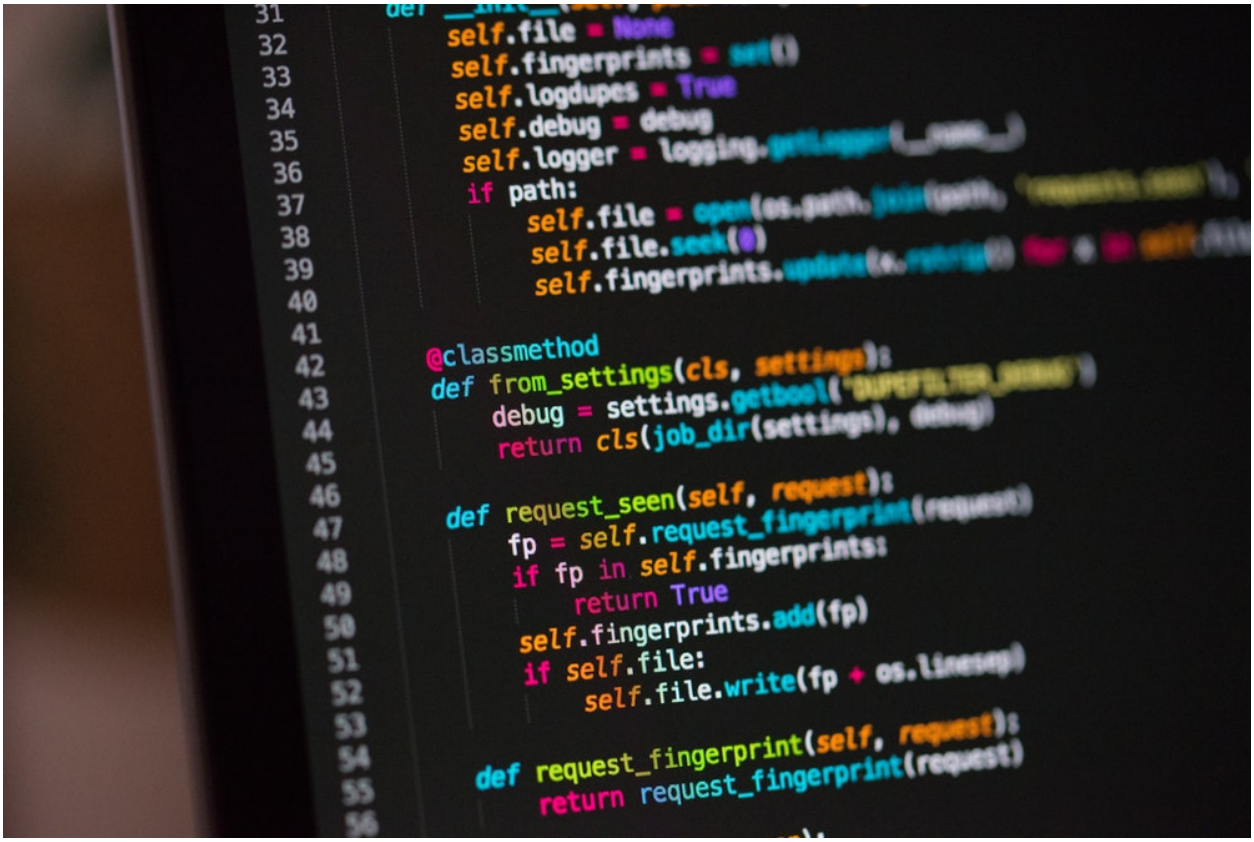
INP ENSEEIHT

Rapport intermédiare



Hamza Mouddene
Manal Hajji

10 décembre 2020

1 Les services Hidoop

1.1 La classe workerImpl

Le service Hidoop fournit le support pour l'exécution répartie. Un démon (Worker) doit être lancé sur chaque machine. Nous avons utilisé RMI pour la communication entre ce démon et ses clients. Nous avons fait une implèmentation de Worker , WorkerImpl qui a comme but de lancer un runmap qui lance la méthode map. A la fin du traitement, on utilise callback pour prévenir la fin de notre traitement et on ferme les deux fichiers reader et writer. Dans le main, on crée le serveur lié au port entré par l'utilisateur en ligne de commande avec LocateRegistry.createRegistry(port).

1.2 La classe Job

Nous avons implémenté l'interface JobInterface en écrivant startJob qui applique le runMap sur chaque machine du cluster à l'aide des éléments nécessaires (path to the file ,reader le fichier source sous le format line, writer fichier pour le résultat sous la forme key value). On obtient alors le nombre de fragment à l'aide du nom du fichier. L'application sur tous les fragments se fait selon si on a 1 fragement , ou plusieurs mais le principe reste le même. Le but est alors d'appliquer runmap avec les workers disponibles dans la ligne au dessus .

1.3 CallBack

Dans cette interface, nous avons deux fonctions taskDone c'est une opération bloquante qui nous permet d'attendre que le traitement sur tous les fragments soit fini , en utilisant ici un sémaphore qui dans le cas où on a nbfragments = compteur on fera une opération release sur le sémaphore , et qui nous permettra dans ce cas de connaitre si il y'a une fin de la tâche, et une autre fonction getTaskDone qui retourne le Semaphore en question .

1.4 La classe HidoopClient

C'est notre lanceur,il faut lui donner java HidoopClient fichier format nbmachines ; un fichier écrit sur HDFS via HDFSClient ; le nom du fichier traité local ; un format et le nombre de machines dans notre cluster. On initialise les objets utiles : Registre, MapReduce et Job. Nous avons introduit une liste (urlworkers) qui contient les url reliées aux démons des machines de notre cluster. Dans un try-catch on initialise les variables, si ils sont valide on crée le registre sur le port indiqué. On fait ensuite un Naming.lookup sur chacun des url des démons de chaque machine du cluster. On crée une instance des objets Job et MapReduce. On lance un startJob, on attend que toutes les tâches soient terminées, on récupère le fichier qui vient d'être traité avec reduce (à l'aide de HdfsRead).