Billy Sayasaeng

09 158 699

PROJET

Processus concurrents et parallélisme

IFT 630

Travail présenté à

Gabriel Girard

Baccalauréat informatique

Université de Sherbrooke

Jeudi 26 avril 2012

Sommaire

Fonctionnement du programme

Le programme prend en paramètre un argument. Avant d'exécuter le programme, on doit spécifier un mot de passe à crypter (ex. : Salut). Le mot de passe peut être des lettres en majuscule ou en minuscule et des chiffres. L'exécution du programme se fait par la fonction main dans la classe *Master.java*, il va prendre la chaine de caractère passé comme argument et il va le crypter avec l'algorithme md5.

Par la suite, on spécifie si on veut l'exécuter en mode local (C) ou externe (E). Le mode local vous permet de créer des fils d'exécution de travailleur sur la même machine. Le mode externe permet l'exécution du travailleur sur plusieurs postes de travail différent pour permettre l'expédition des tâches. Si l'option E est choisie, vous devez rentrer l'adresse IP du maître.

Ensuite, le programme affiche la chaine de caractère crypté par l'algorithme de cryptage md5. On doit entrer le nombre de travailleurs à dispersé pour le calcule de décryptage.

Le programme demande d'entrer le nombre de mots de passe à tester total. Par exemple, si on entre 5 travailleurs et 100000 mots de passe à tester, chaque travailleur testerait 20000 mots de passe différents. Par la suite, chaque travailleur sous-divise le 20000 par un nombre de thread sur la machine du travailleur. J'ai fixé 10 threads pour mon programme.

Une fois que le maître termine la séparation des mots de passe à tester, il demande d'enter l'adresse IP des travailleurs pour leur envoyer un message contenant les informations requises pour faire les tests de mots de passe.

Finalement, le maître attend qu'un travailleur lui retourne le mot de passe décrypté.

Pour les postes de travailleur, le programme doit exécuter la fonction main dans *WorkerMain.java.* Il demande d'entrer l'adresse IP du serveur et il attend pour recevoir le message contenant les informations pour faire les tests.

Développement du projet

Le langage de programmation utilisé pour le développement du projet est java. Java inclut déjà l'algorithme md5 dans la librairie *java.security.MessageDigest*. L'objet *MessageDigest* transforme, avec md5, le mot de passe en *array* d'octets et le programme converti l'*array* en une chaine de format hexadécimal. La classe MD5Standard a une fonction qui crypte une chaine de caractère passé en paramètre.

La classe Master s'occupe de créer les limites de test pour chaque travailleur. Dans le mode locale, il s'occupe de créer et d'initialiser les travailleurs. Il attend de recevoir un message d'un travailleur qui a trouvé le mot de passe crypté et il

interrompt les autres travailleurs par la suite. Dans le mode externe, il envoie les messages aux travailleurs et attend de recevoir par message le mot de passe trouvé.

La classe WorkerMain s'occupe de recevoir les informations du maître. Il divise la tâche pour les donner au travailleur. Il exécute par la suite plusieurs travailleurs.

La classe Worker teste les mots de passe possible, une par une, selon la limite de chaine de caractère imposé par la classe Master. Il crypte une chaine de caractère, la compare avec le mot de passe crypté et si c'est identique, alors il a trouvé le mot de passe. Si le travailleur a trouvé le mot de passe, il envoie un message au maître pour lui dire qu'il a trouvé le mot de passe crypté.

La classe StringIterator fait itérer une chaine de caractère entré en paramètre. L'ordre de l'itération est basé sur la définition d'une variable de chaine de caractère nommé *tableau*. L'itération d'une chaine de caractère se fait par le positionnement du mot de passe à tester et l'accumulation du dernier caractère avec la prochaine position du caractère a testé. Le principe est le même que l'addition en mathématique. Si le prochain caractère n'existe pas, on remet le caractère de la position initiale à zéro, et on fait itéré le caractère à la prochaine position, ainsi de suite, de façon récursive.

Idée et amélioration possible

La partie locale ne fonctionne plus. Cela a été implanté que pour tester le fonctionnement des threads et mon but principal était de faire la distribution de tâches sur des ordinateurs travailleurs. Faire fonctionner les deux options serait une amélioration possible pour mon projet. De plus, une amélioration possible pour mon projet serait de faire une liste de serveur et de lire dans cette liste qui contient les adresses IP du maître et des travailleurs pour automatiser le processus. Par contre, il faudrait mettre à jour ce fichier à chaque fois que l'on veut ajouter, supprimer ou modifier la liste des postes. Il serait aussi possible d'améliorer le code derrière l'algorithme de cryptage md5 en codant l'algorithme, soit même.