Rapport sur le projet UGEGreed

DEBATS Julien - LY-IENG Steven

Contents

1 Introduction

1.1 Présentation du projet

Le but du projet UGEGreed est de réaliser un système de calcul distribué au dessus du protocol TPC. L'idée est d'aider les chercheurs qui veulent tester des conjectures sur un très grand nombre de cas en distribuant leurs calculs sur plusieurs machines.

1.2 Présentation de l'application

Typiquement, un chercheur va vouloir tester une conjecture sur tous les nombres de 1 à 1 000 000 000. Pour cela, il va écrire une fonction Java qui teste cette conjecture pour un nombre n donné. Il ne lui reste qu'à exécuter cette fonction sur tous les nombres de 1 à 1 000 000 000. Cela peut prendre beaucoup de temps et on voudrait pouvoir accélérer le processus en partageant la vérification sur plusieurs machines. Par exemple, si l'on dispose de 10 machines, une machine peut vérifier les nombres de 1 à 100 000 000, une autre de 100 000 001 à 200 000 000, etc...

Le but des applications que vous allez développer sera de pouvoir se connecter les unes aux autres pour se répartir les tâches à faire. Ensuite, les applications téléchargeront le Jar correspondant et exécuteront le code pour chacune des valeurs qui leur ont été attribuées et renverront les réponses vers l'application qui a proposé la tâche initiale. C'est elle qui créera le fichier contenant toutes les réponses.

Nous vous donnerons le code nécessaire pour instancier une classe contenue dans un Jar. Ce n'est pas une difficulté du projet.

1.3 Détails sur le fonctionnement de l'application

Quand on démarre une application, on lui donne un port d'écoute sur lequel elle acceptera la connexion d'autres applications. On peut, en plus, donner l'adresse d'une autre application. Dans ce cas, l'application va commencer par se connecter à l'autre application. Si l'application est démarrée sans l'adresse d'une autre application, on dit qu'elle est démarrée en mode ROOT.

Par exemple, on peut démarrer une application en mode ROOT à l'adresse A sur le port 6666, puis une application à l'adresse B sur le port 7777 en lui disant de se connecter à A sur le port 6666, et enfin une autre application à l'adresse C sur le port 8888 en lui disant de se connecter sur l'application à l'adresse B sur le port 7777. Les trois applications forment un réseau qui vont pouvoir s'échanger des informations.

Une fois qu'une application est démarrée, l'utilisateur va pouvoir demander le test d'une conjecture en donnant l'url du Jar, le nom qualifié de la classe contenue dans le Jar et la plage des valeurs à tester. Les calculs à faire doivent être répartis entre les différents membres du réseau et les réponses doivent être collectées par l'application qui a fait la demande. De plus, les clients doivent pouvoir moduler la charge de travail qu'ils acceptent. Par exemple, si le client fait déjà beaucoup de calculs pour le réseau, il doit pouvoir refuser.

De plus votre protocole doit permettre à une application qui n'a pas été démarrée en ROOT de se déconnecter du réseau sans impacter le reste du réseau, ni perdre des calculs. En particulier, si cette application s'était engagée à faire des calculs, ces calculs devront être réalisés par d'autres applications du réseau après son départ.

Sauf en cas de déconnexion d'une autre application du réseau, les applications ne doivent pas initier d'autres connexions que la connexion qu'elles établissent au démarrage

2 UGEGreed

2.1 RFC

Vous pouvez consulter notre RFC à tout moment ici

2.2 Code

Nous avons commencé à coder le projet en travaillant sur le Readers, que nous avons pensé grâce à notre RFC, il y aura donc un URLReader, INTreader et un LONGReader. Ensuite, nous avons décidé de commencer à l'aide des derniers TP de Réseau la partie connexion entre les applications. TODO

3 Utilisation

3.1 Connexion

TODO

Pour pouvoir lancer le code, il faudra se rendre au niveau du //jar// et lancer la commande

Pour la root

```
java -jar Application.jar <Host> <Port>
```

Pour la une autre application

```
java -jar Application.jar <Host> <Port> <ServerHost> <ServerPort>
```

3.2 Déconnexion

TODO

3.3 Lancement

TODO

4 Difficulté rencontré

4.1 Difficulté sur la RFC

Au début n'ayant pas le sujet assez clair, nous avons commencé à faire une RFC qui n'était pas du tout en rapport avec le sujet du projet, cependant à l'aide de Monsieur Carayol, nous avons mieux appréhendé le sujet et avons pu partir sur de bonnes bases.

Cependant, lors de la soutenance de la RFC, il nous a été montré que notre RFC n'était pas vraiment ce qui était attendu, mais rien de grave, c'étaient quelques petites choses à corriger. Nous avons alors demandé à notre enseignant de plus amples explications pour mieux cerner ce qui était demandé et améliorer notre RFC, aussi pendant ce temps nous avons essayé de commencer à travailler sur la partie code du protocole.

4.2 Difficulté sur le code

Au début du projet, nous ne savions pas exactement comment commencer, mais après avoir pataugé un peu, on a essayé plusieurs choses ce qui nous a permis de nous débloquer. Aussi, nous avons eu un problème de compilation lorsque l'on devait utiliser la classe des Tables de Routages dans la classe principale, la classe de routage n'était pas reconnue lors de la compilation et après avoir fait plusieurs tests et demandé au professeur, il s'avérait que le problème ne venait pas de nous, mais de l'ide Eclipse, car du côté de notre enseignant tout avait marché alors qu'il utilisait Intelij.

5 Division du travail

Pour ce qui est du partage du travail nous avons partagé assez équitablement, on a essayé de faire un maximum pour éviter que la charge de travail repose plus sur une personne qu'une autre.

5.1 RFC

Pour la RFC, on a beaucoup travaillé de notre coté, mais on faisait des points pour assez vite comprendre la situation et échanger les idées entre nous et nous mettre d'accord dessus ou non.

5.2 Code

À ce niveau, on essayait d'avancer chacun de notre côté lorsque l'on pouvait, mais surtout, on a essayé un maximum de coder ensemble sur Eclipse à l'aide de l'extension CodeTogether comme ça, il était assez simple de savoir assez rapidement d'où venais l'erreur et on pouvait intervenir plus rapidement, lorsque l'on travaillait ensemble sur CodeTogether, on essayait d'envoyer un peu prêt le même nombre de commits chacun sur le git

6 Annexe

6.1 Commentaires

On a trouvé le projet de réseau plutôt intéressant, un projet qui nous laissait beaucoup de liberté, où il fallait poser beaucoup de questions pour pouvoir se donner une ligne directrice, même si le projet était assez complexe et nous pensions que le projet serait très très long, nous avons pris du plaisir à travailler dessus.