Rapport sur le projet UGEGreed

DEBATS Julien - LY-IENG Steven

UgeGreed

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 Architecture

Notre architecture se résume surtout a l'envoi de plusieurs types de trames pour pouvoir tout faire.

2.1 Transmissions des données

Pour nous la transmission des données se fait par plusieurs types de trames

Connexion

Lors de la connexion d'une application au réseau, l'application parente a cette dernière (celle a qui elle s'est connecté) transmet une trame NEW LEAF qui stocke l'adresse de l'application et son adresse, et on fait remonter cette trame jusqu'à la ROOT tout en ajoutant les applications par lesquels la trame est passé, en plus de cela, chaque application devra en même temps mettre à jour sa table de routage en conséquence.

La ROOT transmettra aussi cette trame vers ses applications enfants qui ne lui ont pas envoyé cette trame pour pouvoir mettre à jour leurs table de routage tout en ajoutant aussi à leurs tours leurs adresse à la fin de la trame.

Déconnexion

Lors d'une demande de déconnexion l'application va envoyer une trame signalant à toutes les applications du réseau sa mise en déconnexion pour qu'elles suppriment cette application de leurs table de routage mais aussi pour que les applications connexes se connectent entre eux pour délaisser l'application qui se déconnecte.

Lancement de la tache

L'application sur laquelle on initie la tâche va pinger toute les applications du réseau pour savoir lesquels sont libre, celles qui seront libres enverrons un ping réponse disant si elles sont libre ou non. Celle qui seront libre se feront attribuer une partie de la tache.

Lorsque l'on reçoit une tâche, via une trame, on regarde l'URL ou bien le chemin par lequel on doit passer pour accéder au jar et aussi on prend en compte le fichier dans lequel doit être sauvegardé les données.

A chaque fois que l'on doit fait une conjecture, le résultat sera directement mis dans le fichier. Une fois la tache fini l'application se met en attente d'une nouvelle tache

3 UGEGreed

3.1 RFC

Vous pouvez consulter notre RFC à tout moment ici

3.2 Code

Nous avons en majorité utilisé la RFC pour faire la partie code même si il y a certains moment ou nous avons décidé de dévier légèrement de la RFC. Mais globalement cela reste la même chose que ce qu'on avait prévu au départ. Il y a des choses que nous avons ajouté ou supprimé selon le besoin.

4 Utilisation

4.1 Connexion

TODO

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

Pour la root

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

ou bien

```
java bin.fr.uge.greed.Main <Host> <Port>
```

Pour la une autre application

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

ou bien

```
java bin.fr.uge.greed.Main <Host> <Port> <ServerHost> <ServerPort>
```

Connexion établis

Lorsqu'une connexion s'etablis la table de routage et la table de connexion se met a jour et nous affiche

```
-------------Table of connexions--------------
Conneted from : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:4444 remote=/127.0.0.1:51609]

-----RouteTable------
/127.0.0.1:5555 : /127.0.0.1:5555
```

Apres une nouvelle connexion d'une application à l'application qui se trouve a l'adresse 4444

```
-------------Table of connexions--------------
Conneted from : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:4444 remote=/127.0.0.1:51609]
Conneted from : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:4444 remote=/127.0.0.1:51612]

-----RouteTable------
/127.0.0.1:5555 : /127.0.0.1:5555,
/127.0.0.1:6666 : /127.0.0.1:6666
```

(Ceci est un exemple lorsque l'on connecte deux application à une autre)

- le "connected from" s'affiche pour présenter les applications qui se sont connecté à cette application

- le "connected to" s'affiche lorsqu'on se connecte à une autre application, il ne peut y avoir que un ou 0 "connected to"

4.2 Déconnexion

Pour pouvoir se déconnecter "correctement sans "ctrl+c" il y aura un scanner sur chaque applications qui lorsqu'il lira

```
DISCONNECT
```

déconnectera l'application

Sur le terminal qui se déconnecte on aura

```
---------------------
Disconnecting the node ...
avr. 10, 2023 4:10:24 PM fr.uge.greed.Application lambda$1
INFO: Disconnected Succesfully
---------------------
```

et ça nous renverra sur la ligne de commande

Pour les autres terminaux

```
Connexion closed >>>>>>>>>>>>>>>>>>>>>>>>>> /127.0.0.1:51609

-------------Table of connexions--------------
Conneted from : java.nio.channels.SocketChannel[connected local
    =/127.0.0.1:4444 remote=/127.0.0.1:51612]

-----RouteTable------
/127.0.0.1:5555 : /127.0.0.1:5555
```

4.3 Lancement

Le lancement d'une tache se fera en utilisant la commande:

```
START <URL/JAR> <Nom classe> <Debut> <Fin> <Fichier resultat>
```

5 Évolutions depuis la Beta

Lors de la soutenance Beta notre code contenait pas encore les tout les reader, nous et nous partions sans le savoir sur un serveur bloquant. Nous n'avions non plus pas le lanceur de tâches

5.1 Maniere de traiter

6 Fonctionnalités

6.1 Actives

- La connexion des applications entre elles.

- Reconnexion d'une application vers une autre avec mise à jour partielle de la table de routage.

6.2 Active Partiellement ou pas implémenté

- La transmission des données est fonctionnel mais pas correctement implémenté.

- La table de routage n'est pas encore complètement implemenété, il nous faut verifier que via les trames transmises, la table de routage de toutes les applications sont mis a jours correctement, cependant la table de routage de l'application avec les applications connexe est bonne.

- Nous avons fait la partie traitement du JAR et lancement des fonctions, cependant ce n'est pas encore connecté au reste de l'application

6.3 Non Active

- De souvenir rien

7 Difficulté rencontré

7.1 Difficulté sur la RFC

Au début n'ayant pas le sujet assez clair en tête, 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. (On précise que ceci c'est passé bien avant la soutenance)

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.

7.2 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 notre terminal, en effet nous utilisions un terminal Windows pour compiler et lancer le code. Nous avons réglé notre problème en ajoutant l'option -cp lors du démarrage des applications

Un autre problème que nous avons rencontré mais que nous nous y attendions déjà était le fait de reconnaître de quel socketChannel à été envoyé la trame. Nous avons décidé de changer un peu par rapport a la rfc et d'ajouter l'adresse qui a envoyé la trame et en regardant la table de routage on regardais de quel valeur venait l'adresse.

La table de routage aussi a fait partis des difficulté de ce projet, en effet lorsque l'on supprimais les connexions la table de routage ne se mettais pas à jour. elle ne se mettais à jour uniquement si on ajoutais des éléments, on a cependant réussi, au début on pensait utiliser des trames et envoyer pour mettre a jour la table de routage pour les applications à proximité mais nous avons trouvé une méthode pour le faire sans, cependant la table de routage des autres applications doivent être modifier pour supprimer l'application qui s'est déconnecté.

Après la soutenance Beta nous avons rencontré un problème majeur qui provenait des trames envoyé mais que nous avons essayé de régler, cependant lorsque nous avons soit disant réglé le problème des trames, nous avons eu un problème d'envoi des paquets, en effet les paquet s'envoyaient parfois mais parfois ne s'envoyaient pas. Nous pensons avoir réglé ce problème mais maintenant à par intermittence le terminal qui est sensé recevoir la trame boucle sans rien récupérer et est en mode write.

Malgré beaucoup de commits sur le git et de test, nous avons passé beaucoup de temps a débugger le code, une erreur pouvait en entrainer une autre et une erreur pouvait aussi quand elle était résolue en cacher une autre. Pour pouvoir débugger nous regardions a chaque fois les problème qu'affichait le terminal mais quand on ne trouvait vraiment pas, Monsieur Carayol nous a bien prêté main forte.

8 Division du travail

A propos de la division du travail entre binôme, cela à été assez équitable. On a fait le maximum pour éviter de trop charger une personne par rapport a l'autre. On essayait de travailler environ 3h par jours si ce n'est plus sur le projet weekend compris.

8.1 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.

8.2 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 commit chacun sur le git. Cependant lorsque les stages ont commencé il nous à été un peu plus compliqué de nous voir en après-midi, l'un n'était pas encore en stage tandis que l'autre si, on ne pouvait donc se voir que le soir ou le dernier weekend du rendu de projet. ce qui à amener a ce que celui qui n'etait pas en stage de travailler légèrement plus que l'autre.

8.3 Rapport

Le rapport à été mis à jours par une seule personne, cependant l'autre binôme donnait des points qu'il fallait changer ou qu'il fallait ajouter.

9 Annexe

9.1 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.

9.2 Remerciement

Nous remercions l'équipe chargé de la matière Réseau pour le projet intéressant qui nous à été fournis mais aussi pour tout le soutient et idées que vous nous avez données tout le long du projet.