Programmation Réseau Edouardo Rodrigues
Request for Comments: 1 Paul Ancel
UGEGreed 8 Mai 2023

Auteurs: Edouardo Rodrigues & Paul Ancel

Titre: UGEGreed

Date: 8 Mai 2023

Request for Comments: 1

CONTENU

INTRODUCTION

INTRODUCTION

Le projet UGEGreed est un système de calcul distribué conçu pour aider les chercheurs à tester des conjectures sur un très grand nombre de cas en distribuant leurs calculs sur plusieurs machines. Le but de ce système est d'accélérer le processus de vérification en partageant les tâches de calcul sur plusieurs machines, ce qui permet d'obtenir des résultats beaucoup plus rapidement.

Les calculs à faire sont alors répartis entre les différents membres du réseau, et les réponses sont collectées par l'application qui a fait la demande.

Le protocole mis en place permet également aux clients de moduler la charge de travail qu'ils acceptent et de se déconnecter du réseau sans impacter le reste du réseau ni perdre des calculs. En particulier, si une application qui s'était engagée à faire des calculs quitte le réseau, ces calculs doivent être réalisés par d'autres applications du réseau après son départ.

Il est à noter que 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.

En résumé, UGEGreed est un système de calcul distribué qui permet d'accélérer le processus de vérification des conjectures en distribuant les tâches de calcul sur plusieurs machines. Le protocole mis en place permet une gestion efficace des tâches de calcul, une répartition de la charge de travail et une gestion des déconnexions sans impact sur le reste du réseau.

1. Démarrage d’une application

1.1 Définir un port d’écoute

Avant de pouvoir démarrer une application, il est nécessaire de définir un port d'écoute sur lequel l'application va attendre les connexions entrantes. Le port d'écoute doit être choisi de manière à ne pas entrer en conflit avec d'autres ports utilisés par des applications sur la même machine.

Une fois le port d'écoute choisi, l'application peut être lancée et se mettre en attente de connexions entrantes.

1.2 Définir un port de connexion

On peut démarrer une application en lui fournissant un port de connexion, en plus du port d'écoute, cela va permettre à l'application de se connecter à l'autre application en premier lieu. Une application à un parent et peut avoir plusieurs fils.

1.3 Démarrage en mode ROOT

Une application démarré en mode ROOT est une application qui démarre en premier dans le réseau et qui n'a pas de port de connexion fournie, seulement un port d'écoute. Elle n'a donc pas de parent et peut accepter la connexion de plusieurs clients (peut avoir plusieurs fils).

1.4 Mode de communication

Le protocole utilisé pour la communication de ce réseau sera en TCP.

Chaque trame aura une id en départ qui est donc un entier suivi des données correspondantes à cet id.

id = 0, pour la plage à tester ainsi que l'URL

La borne_inf et borne_sup sont incluses.

0 (entier)	src (entier)	dst (entier)	id (entier)	borne_inf (entier)	borne_sup (entier)	URL (encodée en UTF-8)

id = 1, pour un résultat à router vers son application d'origine.

On aura ici dans les données trois entier qui correspondent à la source, la destination qui sont des ports, l'id de la conjecture ainsi que le résultats encodée en UTF-8.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|1 (entier) | src (entier) | dst (entier) | id (entier) |
resultats (encodée en UTF-8) |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

id = 2, pour une déconnexion

On aura ici dans les données un entier qui correspond au port auquel l'application devra se connecter.

```
- - - - - - - - - - - - - - - - - - -
|   2 (entier)  |  port (entier)  |
- - - - - - - - - - - - - - - - - - -
```

Tous les ids ainsi que valeur entière sont en big endian, et le reste sera donc encodé en UTF-8.

2. Récupération de la demande des chercheurs

2.1 Format de la demande

La demande des chercheurs est sous un format bien précis :

- La conjecture a tester (chemin classe et nom de la classe)
- La plage (range) a tester
- Nom du fichier de collecte des résultats

Le format de la demande doit être précis et strictement respecté afin de garantir un traitement efficace et sans erreur de la demande. Une fois le format de la demande établi, les applications peuvent procéder à la récupération de cette demande.

2.2 Récupération de la demande

Une demande se récupère grâce à une URL que l'application qui demande la conjecture va récupérer et diffusé ensuite à chaque application dont elle demandera une conjecture et ainsi de suite. Chaque membre du réseau disposant de l'URL va pouvoir récupérer cette demande.

3. Lancement des tests de conjectures

3.1 Définition des sous-plages

L'application en mode ROOT va définir des sous plage en essayant d'équilibrer le rapport entre une charge de travail raisonnable pour un client ainsi qu'une charge de travail équitable pour chaque client fils et ainsi de suite (les fils avec leurs fils). La définition des sous-plages est libre tant qu'elle respecte le protocole de transmission des données.

3.2 Lancement des tests

Une fois un client initié, il va donc être en charge de récupérer la conjecture grâce à l'URL fournie par son parent, et de faire tourner cela avec la sous-plage également fournie par son parent avec qui elle communique.

3.3 Déconnexion d'un client

Il y a 3 cas de déconnexions :

- Si une application décide de se déconnecter du réseau alors qu'elle n'a pas de tâches en cours, elle peut simplement fermer sa connexion avec les autres applications et quitter le réseau.

- Si une application à des tâches en cours mais n'a pas encore envoyé ses résultats aux autres applications, elle doit d'abord envoyer ses résultats en attente avant de se déconnecter. Une fois que tous les résultats ont été envoyés, elle peut fermer sa connexion avec les autres applications et quitter le réseau.

Une application qui se déconnecte se chargera d'envoyer toutes les informations nécessaires à ces voisins afin de garder le réseau relié.

4. Résultats

4.1 Constitution des résultats par un membre du réseau

Chaque client du réseau ayant fait des calculs devra rooter ou collecter les résultats dans un fichier selon l'application qui à demander la conjecture.

4.2 Assemblage des résultats

L'assemblage des résultats survient lorsque l'application qui a demandé la conjecture reçoit des trames spécifiques à la collection des résultats. Celle-ci
créera alors un fichier contenant tous les résultats.

4.3 Publication des résultats

La publication des résultats est donc fait par l'application qui à demandé les conjectures.