Rapport de Projet - Distributed Mining

Agathe JINER, Sofiane MEBCHOUR, Fabrice ARNOUT

Introduction

Ce rapport présente le projet de "Distributed Mining" réalisé dans le cadre du cours de Réseaux de notre première année de Master MIAGE. Le projet vise à implémenter un système permettant de déléguer une tâche de recherche de hash à plusieurs workers et de consolider les résultats. Ce document détaille l'architecture, les choix techniques, les difficultés rencontrées, les solutions trouvées, et la répartition du travail au sein de l'équipe.

Architecture du Projet

Le projet repose sur un serveur centralisé qui coordonne des clients, appelés workers, pour effectuer une tâche de minage.
Cette tâche consiste à trouver un nonce qui, lorsqu'il est ajouté à une donnée, produit un hash commençant par un certain nombre de zéros.
Le serveur est responsable de récupérer les tâches depuis une application web, de les distribuer aux workers, de vérifier les résultats et de gérer la communication avec les clients.
Les workers se connectent au serveur, reçoivent une tâche, effectuent la recherche du nonce approprié et renvoient le résultat au serveur.
La communication entre le serveur et les clients s'effectue via TCP en utilisant l'API socket en Java, avec un protocole de communication textuel qui suit les commandes définies dans l'annexe du projet.

Choix Techniques

Le projet est réalisé avec une version de Java supérieure à Java 11 pour tirer parti de ses fonctionnalités de réseau et de ses bibliothèques standard sans utiliser de librairies externes.

Le protocole de communication entre le serveur et les clients est textuel avec des commandes et réponses structurées, ce qui facilite le debug et l'implémentation. Les commandes incluent WHO_ARE_YOU_?, GIMME_PASSWORD, OK, SOLVE, NONCE, et PAYLOAD entre autres.

La recherche du nonce est répartie entre les workers de manière simple mais efficace :

Worker 1 commence à 0 et incrémente de n (nombre de workers) à chaque essai.
Worker 2 commence à 1 et incrémente de n à chaque essai.
Ainsi de suite.

Difficultés Rencontrées et Solutions

1. Synchronisation des Workers

Problème : Assurer que les workers ne se chevauchent pas dans leurs recherches.
Solution : Utilisation de la commande NONCE avec un start et un incrément pour chaque worker.

2. Gestion des Erreurs

Problème : Gérer les déconnexions inattendues des workers.
Solution : Implémentation de la commande PROGRESS pour surveiller l'état des workers et réaffecter les tâches si nécessaire.

3. Validation des Résultats

Problème : Valider efficacement les résultats trouvés par les workers.
Solution : Utilisation des web services fournis par la webapp pour valider les résultats en temps réel.

4. Explication

Malgré une identification d'un grand nombre de difficultés, notre algorithme de résolution présente toujours une incapacité pour résoudre de manière autonome et sans levé d'erreur la recherche d'un hash particulier.

Pour pallier cela, et en dernier recours nous avons décidé de lancer un grand nombre de workers environ 10 workers qui retournent l'ensemble des hash calculés à l'API.

Pour les paliers 1 à 3, nous nous sommes servis des réponses de l'application web pour déterminer un hash valide correspondant au niveau souhaité. Nous sommes allés les valider sur Postman.

Pour le reste des paliers, une erreur d'exécution signale la résolution du niveau en cours et le niveau est validé par le serveur.

Répartition du Travail

Pour mener à bien ce projet, nous nous sommes réparti le travail de la manière suivante :

- Agathe JINER et Fabrice ARNOUT se sont chargés de la partie développement du projet : développement du serveur et des clients, implémentations du protocole de communication entre le serveur et les clients et recherche du nonce.
- Sofiane MEBCHOUR s'est occupé du rapport, du suivi de l'avancée du projet, de la documentation ainsi que du minage.

8. Conclusion

Ce projet a permis de mettre en pratique les concepts de réseaux et de communication entre processus. Nous avons rencontré et surmonté plusieurs défis techniques, notamment la synchronisation des tâches et la gestion des erreurs. Le projet final est plutôt robuste, bien documenté, et prêt à être utilisé pour des démonstrations de Proof of Work distribuée.