Rapport de projet

DATE : 22/12/2024

Présenté par :

Antoine Marmelat, Lina Pereira,
Dimittri Choudhury

CY-TECH

1. Introduction

Nous avons réalisé ce projet C-Wire dans le but de traiter des données massives de distribution d'électricité, tout en respectant un cahier des charges exigeant. Notre objectif était de garantir la fonctionnalité et la robustesse de notre solution, en optimisant notamment le temps d'exécution et l'empreinte mémoire.

2. Organisation et Méthodologie

Nous avons opté pour une organisation de projet simple :

- Un dossier codeC qui contient les fichiers sources C, le Makefile et l'exécutable.
- Un dossier input pour contenir le fichier CSV à trier.
- Un dossier tests où nous stockons les résultats d'exécution précédent.
- Un dossier graph que nous créons à chaque exécution, mais qui sera vide car nous n'avons pas poursuivi le bonus.
- Un dossier tmp où seront stocker les résultats des exécutions.
- Un fichier c-wire.sh qui contient l'ensemble du script shell.

Concernant la méthodologie, nous avons réalisé des commits réguliers et nous avons surtout communiqué en présentiel pour discuter des grandes décisions (gestion de la mémoire, structure de données, etc.).

3. Répartition des Tâches et Collaboration

- **Antoine** s'est principalement occupé du **script Shell** et d'une partie du code C (**le main**).
- **Lina** a coordonné la communication au sein de l'équipe pour veiller à ce que chacun aille dans la même direction et a pris en charge la rédaction et la validation du fichier ReadMe ainsi qu'une partie du codeC.
- **Dimittri** a développé la partie liée au tri (ordre croissant), assuré le débogage global du code, et rédigé ce rapport.

Nous avons planifié la répartition de manière à tirer profit des compétences et des centres d'intérêt de chacun, tout en nous assurant d'avancer de façon cohérente et organisée. Nous avons de plus mis en place un groupe WhatsApp pour pouvoir communiquer plus facilement à distance.

4. Planning et Gestion du Temps

Notre équipe a travaillé de manière soutenue, avec des commits et validations une ou plusieurs fois par semaine. Nous avons commencé pleinement le projet la semaine du 2 décembre, cela nous a permis d'absorber sereinement les imprévus techniques et de réorienter nos efforts si nécessaire (par exemple, ajuster l'allocation mémoire ou peaufiner le traitement des données).

- **Semaine du 2 décembre :** Découverte du sujet, création du dépôt github, création des premiers fichiers (codeC, c-wire.sh)
- **Semaine du 9 décembre :** Ajout du makefile, main, finition du codeC, finition du script shell.
- **Semaine du 16 décembre :** Correction des dernières erreurs, tests sur les machines de l'école, rédaction du Readme et du rapport.

5. Approche Générale du Développement

Tests et validation

Nous avons principalement utilisé des **tests manuels** sur un fichier CSV plus léger pour vérifier rapidement la validité de notre code. Ensuite, nous avons exécuté le programme sur le **gros fichier** afin de valider les performances et la robustesse. Nous avons de plus placé des exemples d'exécution des différents cas dans le dossier test. Pour le cas lv all, nous avons implémenté une colonne supplémentaire pour la partie max et min qui représente la différence entre la capacité totale et la consommation totale du plus petit au plus grand. Pour ce qui est des autres cas, les données sont triées par capacités croissantes.

Difficultés rencontrées

- La **taille du fichier CSV** (plusieurs millions de lignes) nous a amenés à peaufiner nos opérations de lecture et d'insertion afin d'éviter des temps d'exécution trop élevés. De plus la taille du fichier a posé problème lors des commits sur github, à cause de cela, nous avons due réinitialiser l'historique des commits car nous avions un problème entre la branche actuelle et les branches temporaires.
- Nous avons dû veiller à la **gestion de la mémoire** pour ne pas surcharger la machine lors d'un traitement aussi volumineux.

Optimisations effectuées

Notre effort principal a porté sur le **temps d'exécution**. Le choix d'une structure de données équilibrée et l'utilisation de commandes Shell efficaces se sont révélés décisifs pour conserver des performances satisfaisantes à grande échelle.

6. Limitations Fonctionnelles

Nous avons **implémenté intégralement** les fonctionnalités requises par le cahier des charges. Nous avons **renoncé aux bonus** pour mieux nous concentrer sur la qualité et la performance du socle fonctionnel principal, que nous avons entièrement validé. Cependant le temps d'exécution pour les stations lv peut être un peu long, aux alentours de 20 à 30 secondes.

7. Conclusion

Nous sommes satisfaits du résultat final, qui répond pleinement aux exigences :

- Le code est **robuste** et gère convenablement le gros volume de données.
- L'architecture générale permet des **performances satisfaisantes** lors de l'insertion et du tri.
- Chacun a pu contribuer selon ses compétences, assurant une bonne cohérence d'ensemble et un partage efficace des tâches.

Ceci conclut notre rapport. Nous espérons que notre solution saura répondre aux besoins de l'évaluation et mettra en lumière la solidité de notre travail d'équipe.