Rapport Projet Informatique 3
CyEnedis

Antonin PERALTA
Alexandre Halhoute
Shayma Mokhtari

Sujet C-Wire
PréIng-2 Math-Info options Physiques
MI-1 Tri-G
Année 2023/2024

Sommaire

Gestion de l’équipe

Notre équipe s’est organisée autour de Discord ainsi que de Github, notre but est de nous répartir et de parler du travail effectué via discord (appels vocaux ainsi que messages), cela nous a permis d’avoir une bonne répartition des tâches ainsi qu’un projet toujours utilisable et testable par tous les membres du groupe. De plus, grâce à Github, si nous rencontrions un problème, nous pouvions directement utiliser l’historique afin de remonter les modifications sur un système.

Pour la répartition des tâches, Antonin s’est occupé de la partie traitement des données avec la lecture du fichier CSV pour créer l’AVL et ressortir les données dans un autre fichier CSV. Alexandre s’est chargé de la gestion de l’affichage lors du temps d’attente (pendant le traitement des données), ainsi que de l’écriture de certains commentaires manquants dans le code. Enfin, Shayma s’est occupée du script shell et des vérifications des arguments pour celui-ci.

Difficulté rencontrée

Lors de la création du projet, nous avons rencontré plusieurs problèmes, principalement avec le temps de traitement des données. En effet, lors de nos premiers tests, le traitement du gros fichier (celui de 250 Mo) prenait plus d'une minute. Cela était principalement dû à la manière dont nous récupérons les données pour les convertir en fichier CSV par la suite.

Nous avons donc dû repenser cette étape en utilisant les tableaux dynamiques de C ainsi que des fonctions pour connaître la taille d'un AVL. De même, dans le fichier shell, nous avons dû effectuer plusieurs recherches afin de trouver des moyens de faire des vérifications sur les arguments (par exemple, vérifier que le power plant ID est un nombre).

Le dernier problème qui a été identifié est le tri des données dans les fichiers CSV. En effet, celui-ci prenait ~40 secondes sur nos fichiers d'export (par exemple LV_ALL). Il a donc été décidé de refaire une implémentation du quicksort dans le code C afin d'obtenir un temps de traitement plus proche d'une seconde.

Fonctionnalité bonus

Nous avons ajouté plusieurs fonctionnalités bonus, telles que le diagramme. Dans le cas où nous avons un LV ALL, un diagramme de consommation est généré à partir du fichier “min - max”.

Il y a également l’ajout d’un écran de chargement. En effet, lors du traitement des données, vous pourrez observer un écran affichant “chargement” ou encore “les données sont presque entièrement traitées”. Cela est réalisé grâce à la librairie “pthread”, qui nous donne accès au multithreading. Cela nous a donc permis de mettre en place un affichage dynamique pendant que les données sont traitées.

Nous avons également ajouté plusieurs systèmes nous permettant de tester notre projet de manière plus facile, tels que des tests unitaires avec le fichier “application_test.c” ainsi que “test.sh”, qui nous permettent d’exécuter plusieurs cas de données. De même, nous avons aussi ajouté des fonctionnalités pour vérifier le fonctionnement interne, comme la lecture des paramètres, la validation que l’AVL est bien un AVL et qu’il est correctement construit, etc. On peut également noter la création d’un système de benchmark avec le fichier “benchmark.c”, qui nous permet d’obtenir le temps d’exécution de chaque opération à toutes les étapes du programme. Celui-ci se lance avec la variable d’environnement “RUNNING_BENCHMARK=1”. Enfin, un fichier pour vérifier les fuites de mémoire, “memleak_check.c”, nous permet de traquer tous les appels à malloc dans le programme et de vérifier qu’ils sont bien libérés (free) à la fin du programme.