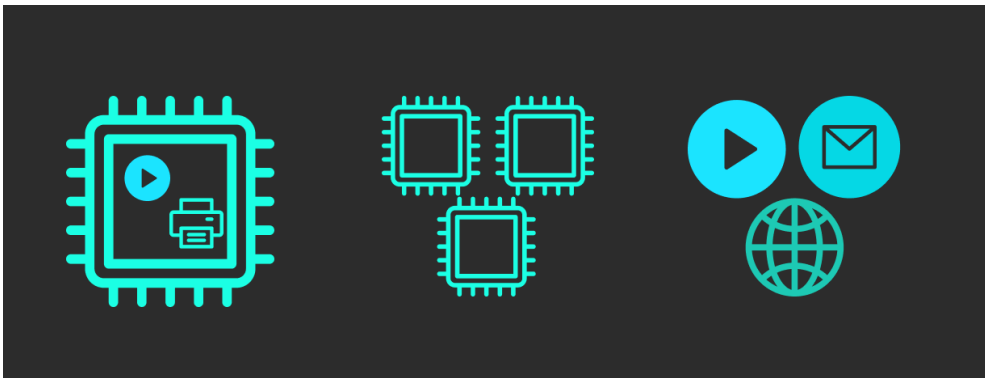
Projet MultiProgramming en Python M1 :

Ajout d'un mécanisme de Multiprogramming

sur un script de transfert de fichiers vers un serveur FTP distant



PIERRE GIRAUD

CÉLESTIN CAPTAL

THOMAS BALDASSARRE

GitHub : https://github.com/Giraud-Pierre/Filezilla_multithreading

Contexte du projet :

Le projet consiste à l'ajout d'un mécanisme de multiprogramming tel que :

-*Du multi-threading*

-*Du multi-processing*

-*De la programmation asynchrone*

Il est donc demandé au groupe de choisir la meilleure solution selon le cas dans lequel il se trouve. Ce mécanisme sera alors ajouté à un programme qui réalise la synchronisation d'un dossier local d'une machine vers un site distant FTP en utilisant un serveur Filezilla. Toute modification dans le dossier local sera affectée sur le dossier FTP avec un délai d'attente.Nous avons pour objectif d'optimiser le programme et par conséquent avoir une plus grande vitesse d'exécution sur les modifications à opérer entre le dossier local et FTP.

Organisation :

Pour nous répartir le travail sur ce projet, nous avons créé un discord pour se partager les différentes idées et modifications que nous avions durant les sessions personnelles. Thomas ayant eu des problèmes de compatibilité avec Filezilla nous avons décidé de diviser le travail de la façon suivante :

- Pierre et Célestin se sont occupés de l'étude de Filezilla et de l'implémentation des fonctionnalités sur le programme.
- Thomas s'est occupé de l'analyse des différentes solutions en étudiant les TP réalisés précédemment.

Pour réaliser ce projet il est nécessaire d'étudier le code sur lequel nous allons ajouter la fonctionnalité de multiprogramming. Nous avons donc utilisé la première séance pour étudier le programme et essayer de comprendre le fonctionnement de Filezilla. Pendant ce temps Thomas a réfléchi aux solutions les plus optimales dans notre situation pour ne pas perdre de temps dans la réalisation du projet.

Les choix faits en terme du « **multiprogramming** » :

Pour savoir quels choix est le plus optimal pour améliorer le projet initial, il était nécessaire d'étudier dans quelles parties le multiprogramming devait être effectué.

Nous avons voulu mettre en parallèle les processus en créant un nombre défini au début du programme de threads. Ceux-ci auront pour objectifs d'établir les connexions nécessaires sur le serveur, et d'effectuer les transfert ou suppression simultanéments pour gagner du temps.

De plus, les tâches à réaliser nécessitent peu de puissance, il n'est donc pas nécessaire de mettre en œuvre une solution de multiprocessing.

Le multithreading peut être plus complexe à mettre en œuvre en raison de la nécessité de gérer correctement les problèmes de synchronisation. Mais dans notre cas, une queue permet d'éviter tous conflits. Nous n'avons pas envisagé de faire de la programmation asynchrone car les threads

travaillent sur les mêmes dossiers cela pourrait donc causer des problèmes de suppressions par exemple.

Pour résumer, nous avons choisi l'optimisation des performances et la simplicité d'intégration du multithreading.

Déroulement du projet :

Nous avons rencontré plusieurs difficultés pendant la réalisation de ce projet, cependant nous sommes tous d' accord pour dire que celle qui nous a le plus mis en déroute est la compréhension de Filezilla et du FTP en général. En effet , pour mettre en place un mécanisme de multiprogramming , il est nécessaire d'appréhender le fonctionnement du FTP. Les sources disponibles ayant souvent plusieurs années d'ancienneté,il était très compliqué de trouver des informations fiables. Cependant certains sites comme Quora , Filezilla-project ou chatgpt nous ont bien aidé pour nos questions restées sans réponse.

Nous avons rencontré un problème avec les logs du code, avant modification l'emplacement du fichier log.cong était en chemin relatifs et non absolu ce qui pouvait engendrer des erreurs selon la configuration du client. En effet, la console cherchant les chemins relatifs dans le dossier où elle se trouve et celle-ci ne se trouvant pas forcément dans le dossier où se trouve le fichier, cela peut résulter en erreur sur le chemin de ce fichier. C'est pourquoi nous avons modifié le fichier logger.py pour trouver le chemin absolu du fichier log.conf, quel que soit l'endroit où il se trouve, tant qu'il est dans le même dossier que les fichiers python.

Dans un premier temps nous avons essayé de créer **un thread par fichier à copier** avec un unique talk to FTP. Cependant le FTP limite le nombre de transferts simultanés par connexions FTP, nous avons donc dû mettre en place **une connexion par copies** en limitant le nombre de thread à mettre en parallèle.

Notre première ébauche de ce système de multithreading n'a pas été très concluante car après calcul de temps entre le programme normal et celui en multithreading, nous avons trouvé 5 secondes pour le premier et 15 secondes pour le deuxième. Nous effectuons une connexion par fichier et la terminons une fois que le transfert de ce fichier est fini. Cela prenait beaucoup de temps et n'était pas plus performant.

Pour contrer cela, nous avons mis en place **un système de queue**. Après avoir établi au lancement les connexions de chaque thread, celles-ci restent actives. Pour que dès qu'un transfert se finisse dans un thread, celui-ci en commence un autre en file d'attente sur la même connexion.

Une fois tous les transferts effectués, la fonction any removal regarde et met en thread les éléments à supprimer en parallèle. Dès que la suppression est terminée, les connexions FTP sont fermées.

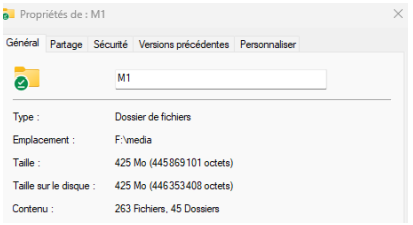
Une solution a été envisagée en conservant les mêmes threads tout du long du programme, cependant par manque de performance cette solution n'a pas été retenue. Elle est trouvable sur la branche GitHub « CreatingThreadsOnlyOnce ».

Résultats :

Tous nos tests sont faits sur un dossier comprenant des sous dossiers avec une taille totale de 4425mo de gros et petits fichiers (de 100mo à 2ko).

Avant en programmation séquentielle :

```
2023-03-08 17:39:08,771 - root - INFO - File created / updated : srv /test/Stages
2023-03-08 17:39:08,795 - root - INFO - File created / updated : srv /test/Stages
Mobilite International - Retour Experience 2022 - Copie.pdf
2023-03-08 17:39:08,811 - root - INFO - File created / updated : srv /test/Stages
2023-03-08 17:39:08,844 - root - INFO - File created / updated : srv /test/Stages
2023-03-08 17:39:08,860 - root - INFO - File created / updated : srv /test/Stages
2023-03-08 17:39:08,876 - root - INFO - File created / updated : srv /test/Stages
2023-03-08 17:39:08,885 - root - INFO - File created / updated : srv /test/Stages
2023-03-08 17:39:08,901 - root - INFO - File created / updated : srv /test/Stages
3.877160787582397
```



On observe un temps de copie de **3.8 secondes** avant multithreading.

Après en programmation parallèle (5 threads) :

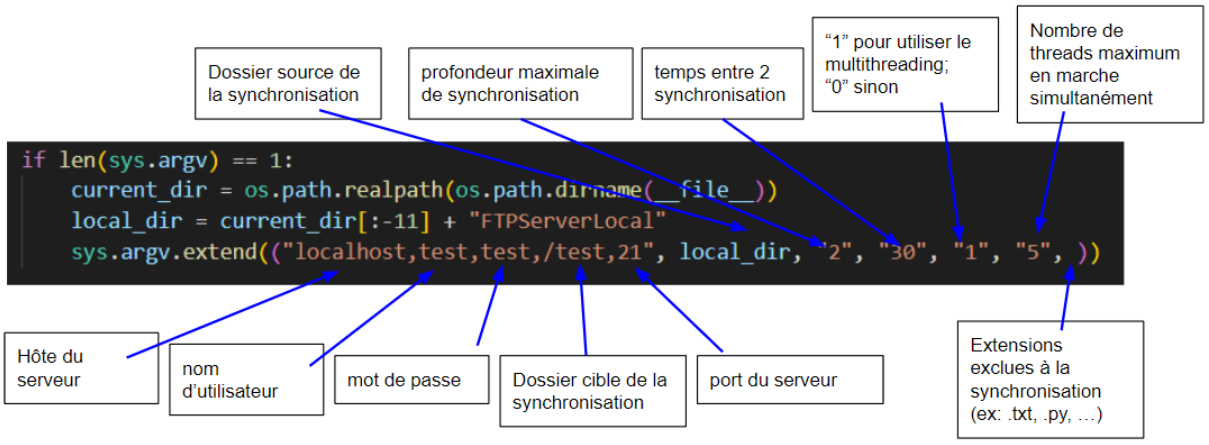
```
2023-03-08 17:36:57,743 - root - INFO - File created / updated : srv /test/Stages M1/GRILLE M1 FR.pdf file GRILLE M1 FR.pdf
2023-03-08 17:36:57,823 - root - INFO - File created / updated : srv /test/ojet hr/HAR-2012.11866.pdf file HAR-2012.11866.pdf
2023-03-08 17:36:58,036 - root - INFO - File created / updated : srv /test/Exploring_Mars.zip file Exploring_Mars.zip
start removing
end removing
1.8019654750823975
```

On observe un réel gain de temps en divisant par 2 le temps de transfert soit un **1.8 secondes**.

Nous avons testé avec différents nombres de thread, plus de 5 threads ne font pas réellement gagner plus de temps de façon significative. De plus, n'utiliser qu'un seul thread fait remonter le temps de transfert de ce dossier à 3.8 secondes ce qui est cohérent avec le programme de base.

Comment utiliser le logiciel:

- Téléchargez le projet sur le github sur la branche main (au format zip)
- Décompressez le projet
- Lancez votre serveur
- Deux méthodes sont alors possibles
    - Remplissez directement les champs par défaut dans le fichier main.py sur votre éditeur de code favori puis exécutez ce fichier main.py



    - Lancez fichier main.py depuis la console avec les arguments appropriés

```
python main.py "localhost,test,test,/test,21" "../FTPServerLocal" 2 30 1 5
```