Description du Programme Multiprocessus

Youness Chaanani

Objectif général

Ce programme a pour but d'exécuter en parallèle plusieurs scripts Python (référencés comme "jobs") en utilisant un nombre spécifié de processeurs. Il journalise également l'exécution et les résultats de ces scripts.

1 Partie II: Question 1

Utilisation

Tout d'abord, je vous invite d'aller le programme principal et remplacez l'URL existant par celui où se trouvent les jobs dans votre machine. Ensuite, ouvrez votre terminal et tapez la commande "python nomdufichier.pp suivi du nombre de processus souhaité". Les résultats s'afficheront à l'URL que vous avez précédemment indiqué. De plus, un fichier de journalisation sera créé pour enregistrer tous les résultats. Des informations supplémentaires, dont la durée d'exécution du programme, seront également affichées dans le terminal, vous trouverez une capture d'ecran qui détait ces modifications.

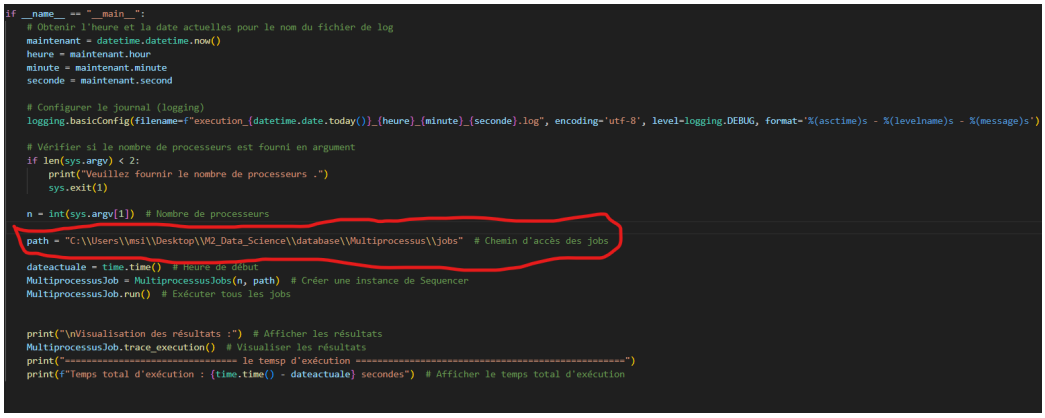

Figure 1: Le lien qu'il faut modifier

Description détaillée

1. **Initialisation** :

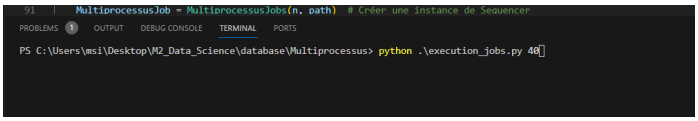

Figure 2: le treminal

- Une classe `MultiprocessusJobs` est définie pour gérer l'exécution en parallèle.
- La classe prend en entrée le nombre de processeurs à utiliser et le chemin d'accès des scripts (jobs).

2. **Exécution des jobs** :
   - Chaque script/job est censé avoir une fonction `run()` qui est appelée pour exécuter le script.
   - L'exécution de chaque job est tentée et les résultats ou les erreurs sont journalisés.
3. **Parallélisation** :
   - Les jobs sont exécutés par lots, chaque lot contenant un nombre de jobs équivalent au nombre de processeurs spécifié.
   - `multiprocessing.Pool` est utilisé pour exécuter les jobs en parallèle.
4. **Journalisation** :
   - Les débuts et fins de lots de jobs sont journalisés.
   - Les résultats de chaque job sont sauvegardés dans un fichier séparé (avec l'extension `.result`) et sont également journalisés.
5. **Visualisation des résultats** :
   - Une fonction `trace_execution` est fournie pour afficher les résultats de chaque job à partir des fichiers `.result`.
6. **Exécution principale** :
   - Lorsque le programme est exécuté, il vérifie d'abord si le nombre de processeurs est fourni en tant qu'argument.
   - Il configure ensuite la journalisation pour sauvegarder les messages de log dans un fichier avec un nom basé sur la date et l'heure actuelles.
   - Enfin, il crée une instance de la classe `MultiprocessusJobs`, exécute tous les jobs, affiche le temps total d'exécution et visualise les résultats.

Étude de l'impact

L'utilisation de plusieurs processeurs permet d'exécuter plusieurs tâches en parallèle. Cela signifie que si vous avez, par exemple, 4 processeurs, vous pouvez exécuter 4 jobs simultanément. Cela réduit le temps total d'exécution, surtout si les jobs sont indépendants et ont des durées d'exécution similaires.

Cependant, l'ordonnancement des tâches dépend de l'OS et de la manière dont il gère l'ordonnancement des processus. En général, l'OS essaiera de distribuer les tâches de manière équitable entre les processeurs.

Il est également important de noter que l'augmentation du nombre de processeurs n'améliore pas linéairement les performances. À un certain point, les coûts associés à la gestion de plusieurs processeurs (comme la synchronisation et la communication entre les processeurs) peuvent commencer à l'emporter sur les avantages de l'exécution parallèle.

Pour étudier réellement l'impact, il serait bon de tester avec différents nombres de processeurs et différents types de jobs..

2 Partie II: Question 2

Instructions pour la Configuration

Pour effectuer la configuration comme indiqué dans la question 1, suivez les étapes ci-dessous :

1. **Modification des fichiers** : Accédez aux deux fichiers, `server.py` et `client.py`. Modifiez-y le lien où se trouvent les jobs.
2. **Lancement des terminaux** :
   - Ouvrez deux terminaux.
   - Dans le premier terminal, saisissez la commande : `server.py "127.0.0.1"`.
   - Dans le second terminal, saisissez la commande : `client.py "127.0.0.1" 40`. Ici, "40" représente le nombre de processus. Vous pouvez choisir le nombre de processus que vous souhaitez.
3. **Considérations importantes** :
   - Assurez-vous que l'adresse hôte (dans cet exemple, "127.0.0.1") est identique dans les deux fichiers.
   - Si la connexion échoue, accédez aux programmes principaux et essayez de changer le numéro de port.
4. **Journalisation** : La procédure de journalisation est la même que celle mentionnée dans la question 1.

Étude de l'impact

Dans la configuration actuelle, j'ai utilisé une seule machine. Cependant, à mon avis, si nous mettons en place plusieurs machines avec un serveur unique pour communiquer avec ces machines, cela pourrait réduire considérablement le temps d'exécution. La raison en est que le serveur pourrait diviser l'ensemble des fichiers en lots distincts, et envoyer chaque lot à une machine différente. Cette répartition parallélisée des tâches permettrait d'accélérer l'exécution du programme principal.