LSINF1252 - Systèmes informatiques 1 : Projet 2

de Bodt Cyril, de Mol Maxime

14 mars 2013

Prérapport

Solution à implémenter

Dans l'optique de répondre aux objectifs du second projet, nous avons imaginé plusieurs solutions. Parmi celles-ci, nous en avons retenu deux, qui se distinguent par leur facilité d'implémentation et leur efficacité. L'objectif de ce prérapport est de présenter ces deux solutions.

Première solution

La première solution que nous avons imaginé est relativement simple à implémenter. L'idée principale consiste à appliquer les filtres séquentiellement sur chacune des images. Ainsi, lorsque que le premier filtre a fini de traiter la première image, il peut passer à la suivante, tandis que la première peut être prise en charge par le second filtre [1].

Cependant, comment comptons-nous traiter un filtre sur une image ? Afin de tirer avantage des différentes threads, nous avons pensé que nous pourrions diviser l'image en plusieurs « bandes », ou tranches de lignes de pixels. Ce nombre de bandes dépend du nombre de thread permis pour le filtre, chaque thread s'occupant d'une bande.

L'avantage majeure de cette solution est quelle évite d'avoir des données partagées entre les threads, ce qui facilite fortement l'imémentation. Grâce à cela, nous avons pu mettre en oeuvre un premier prototype fonctionnel du programme.

Malgré son avantage, cette solution est encore, pensons-nous, perfectible, tout en gardant l'idée principale qu'est le traitement séquentiel des images. Ainsi, nous avons pensé que nous pourrions, dans une bande, appliquer tous les filtres en parallèle. Ceci peut se faire en utilisant des sémaphores : le premier filtre pourrait commencer à traiter sa bande en filtrant quelques lignes [2], au moyen d'un appel à une fonction servant à appliquer un certain filtre sur certaines lignes. Avec un wait avant l'appel à cette fonction et un post apr‘es cet appel, on empêche les autres filtres d'agir sur ces lignes. Ensuite, quand le premier filtre a terminé son travail sur les quelques lignes, un thread du deuxième filtre peut prendre le relais, alors que le thread du premier filtre passe au paquet de lignes de pixel suivant, dans la bande accordée au thread du premier filtre.

Cette extension présente l'avantage de paralléliser davantage le code, mais un problème pratique se pose : comment gérer le fait que les threads des différents filtres sont en nombre différent ? Prenons un exemple : supposons que, pour traiter une image, nous ayons deux filtres à appliquer, disposant de quatre et huit threads. L'image sera divisée en quatre bande, avec un thread du premier filtre sur chaque bande. Or, dans une bande du premier filtre, nous aurons deux threads du deuxième, étant donné que le deuxième filtre dispose de deux fois plus de threads. Comment alors, quand le thread du premier filtre a terminé un premier paquet de ligne, attribuer ces lignes aux threads du deuxième ? Dans l'exemple choisi, nous pourrions décidé de donner la première moitié des lignes à

1. L'ordre d'application des filtres a une importance.
2. Ce nombre de lignes doit être entier.

une thread du second filtre et la seconde moitié à l'autre, mais dans le cas où nous aurions quatre threads pour le premier filtre et cinq pour le deuxième, cela se complique...

Pour résoudre ce problème, nous pourrions décider de considérer que chaque filtre a son propre nombre de bande dans l'image, dépendant de son propre nombre de threads. Il faudrait alors, sur chaque petit paquet de ligne de ligne, placer un ou plusieurs sémaphores, pour contrôler l'ordre d'exécution des filtres et pour permettre, dès qu'un paquet de lignes à été traité par un thread du premier filtre, qu'il puisse être traité par le thread du second se chargeant de la bande dans laquelle se trouve le paquet. Mais cela complique légèrement l'implémentation.

Deuxième solution

Notre deuxième solution consiste à renoncer au traitement séquentiel des images. Nous avons pensé que nous pourrions concaténer les images, de sorte à n'avoir qu'un seul énorme tableau de pixel à gérer. Ensuite, en adoptant une technique similaire à celle présentée ci-dessus, nous pourrions découper l'énorme image en bande pour chaque thread de chaque filtre, avec des petits paquets de lignes à traiter pour paralléliser l'exécution des filtres.

L'avantage de traiter une seule et unique grosse image, c'est qu'en donnant un plus gros travail à chaque thread, on diminue leur temps de vacances (dans le cas où un thread a fini sa bande dans une image seule, il est obligé d'attendre les autres pour passer à la seconde image, tandis que dans la deuxième solution, il pourra finir son travail sur toutes les images). Le désavantage de cette solution est que le tableau regroupant toutes les images sera immense : le copier, le traiter et effectuer des opérations dessus ne sera pas efficace.

Enfin, nous avons également pensé, afin d'améliorer l'efficacité, à essayer de faire en sorte que, lorsqu'un thread a terminé sa bande, qu'il aille aider un autre, encore occupé avec sa propre bande. Mais cela nécessite de la coordination, des mutex sur les paquets de lignes et complexifie le code.

Questions

Pour conclure ce prérapport, nous aurions deux petites questions pratiques concernant l'énoncé du projet :

- Le premier filtre a appliquer sur l'image est-il bien celui situé le plus à l'intérieur des crochets dans les arguments ?
- Dans le cas où plusieurs images sont à traiter, est-ce à notre fonction main de donner au loader les images les unes après les autres ou bien pouvons-nous lui fournir le répertoire entier ?