TP - LIF12 système d'exploitation

Calcul d'image en parallèle

2015/09/15

I Début des threads

II Calcul de fractales

Vous devez paralléliser le calcul d'une image fractale. Vous allez utiliser un modèle où le travail est découpé en petites tâches et où les threads lancés se partagent ces tâches.

Dans ce calcul de fractale, en chaque pixel de l'image, il faut évaluer une fonction de calcul de la couleur. Cette fonction mathématique n'est pas à considérée, elle est déjà écrite dans le fichier mandel−list.c sur le site de l'UE. Afin d'accélérer le calcul, nous vous demandont donc faire éxécuter ce travail par plusieurs threads. Chaque thread faisant un certain nombre de lignes.

Le temps nécessaire à ce calcul n'est pas fixé et dépend des coordonnées du pixel considéré. Il n'est donc pas efficace de donner le même nombre de lignes à chaque thread. Pour faire le calcul, vous devez créer dès le départ un nombre fixé de threads de calcul et une liste des tâches à effectuer. Chaque thread va devoir chercher dans la liste une tâche à faire, effectuer cette tâche et recommencer jusqu'à ce que la liste soit vide.

La liste est un objet partagé et contient uniquement le numero des lignes à traiter. vous devrez faire bien attention à ce que toutes les lignes soient traitées une et une seule fois. Le code préparé contient plusieurs test dont le but est de vérifier cela.

III Travail à faire

Il y a déjà un code séquentiel capable de calculer l'image. Vous devez modifier la fonction main.c pour suivre le shéma décrit dans le paragraphe précédant. C'est à dire :

- Modifier la fonction main pour qu'elle lance 10 threads.
- Faire en sorte que les threads partagent la liste (qui est déjà crée dans le code proposé : lt $*$l = new_...).
- Faire en sorte que les threads de calcul traitent les lignes jusqu'à ce que la liste soit vide (fonction get_tache est prévue pour cela et retourne -1 lorsque la liste est vide).
- Le thread principale doit obtenir le nombre de lignes calculées par chaque thread, faire la somme et vérifier que ce nombre correspond à celui des lignes de l'image.

Attention, la fonction get_ligne n'est pas prévue pour fonctionner en environnement multithread [1]. Vous devez faire en sorte que le programme ne calcule pas plusieurs fois la même ligne, et que toute ces lignes soient bien calculées (si ce n'est pas le cas cela est visible dur l'image obtenue).

1. C'est même le contraire