Rapport LABO1 HPC

Bastien Pillonel

Date : 04.03.2024

Config :

- Lenovo Yoga 7 14ARP8
- Distrib PopOS 64bits (Ubuntu 22.04)
- AMD® Ryzen 7 7735u with radeon graphics × 16

Intro

Dans ce laboratoire le but est de réaliser la partie application de filtre sur une matrice de pixel représenté dans le 1er cas par un vecteur 1 dimension et dans le 2e par une liste simplement chainée. Une fois l'implémentation terminé il est demandé de mesurer et comparé les performances du traitement des deux représentation.

Implémentation

Pas de précision particulière pour la partie traitement 1 dimension. Pour la seconde en revanche, je suis parti de l'idée d'avoir un tableau référencant tout les pixels du bord droit de la sous matrice de même taille que le noyau de convolution utilisé qui serait mis à jour après chaque traitement de pixel individuel. Mon programme commence donc par itérer dans la liste pour trouver ces pixels puis je passe le tableau des reférences de pixels (bord droite) à ma fonction qui applique le produit de convolution. Facile d'utilisation puisque les autres pixel (hors bord droite) sont simplement les next pixels des bords. Cette solution me permet de réduire une bonne partie des itérations si l'on devait à chaque fois retrouver tout les pixels voisins du nouveau pixel à traité.

Résultat

Après quelques ajustage sur le type de mes variables, j'arrive à avoir des images avec les bords détourés.

Analyse des performances

D'abord j'analyse le temps de mon programme en le lançant en argument de l'outil time:

```
time ./lab01 <filename> <filename-edge1/2.png> 1/2
```

Ensuite j'utilise l'outil hyperfine pour faire les mesures. Cet outil est plus avancé que time il permet de lancer plusieur fois la commande et donne plus d'informations statistiques:

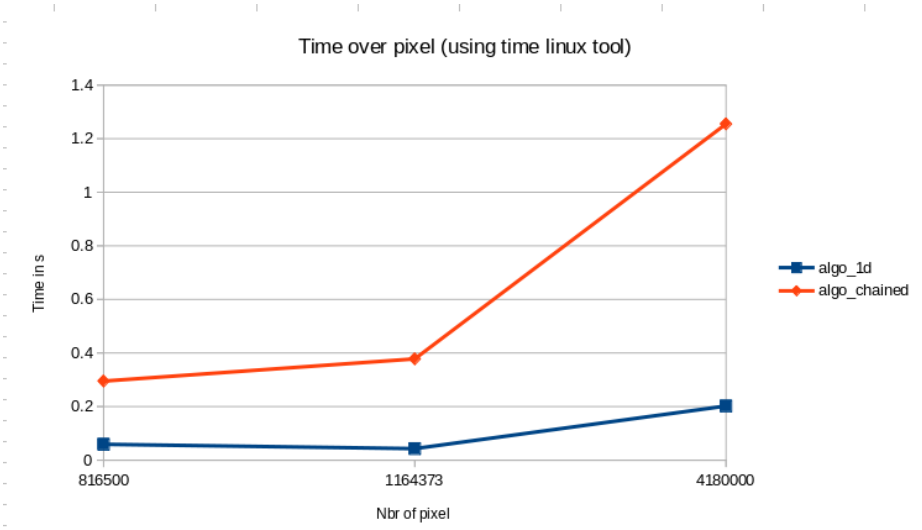
`hyperfine './lab01 <filename> <filename-edge1/2.png> 1/2'`

J'ai ensuite configurer 20 cycle de warmup sur hyperfine pour enlever les outliers qui pourrait trop influencer la moyenne.

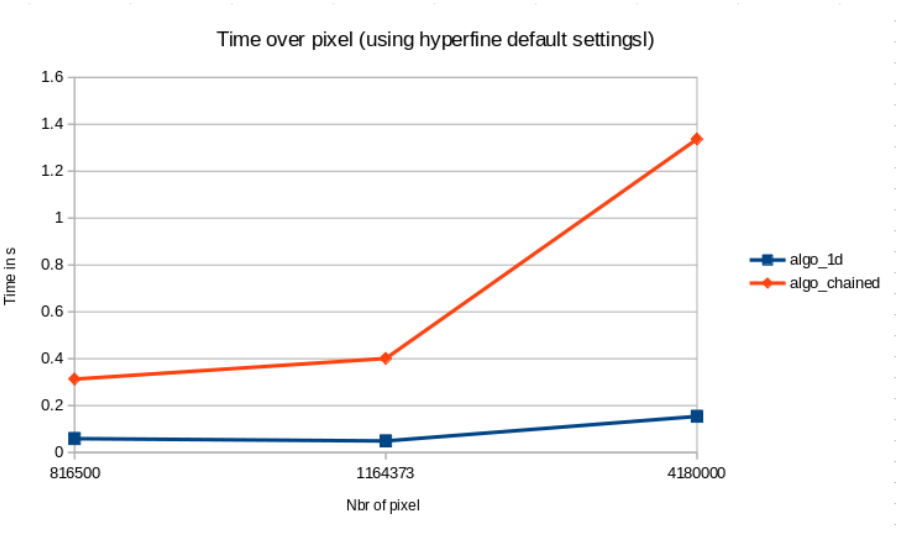
`hyperfine --warmup 20 './lab01 <filename> <filename-edge1/2.png> 1/2'`

Resultats

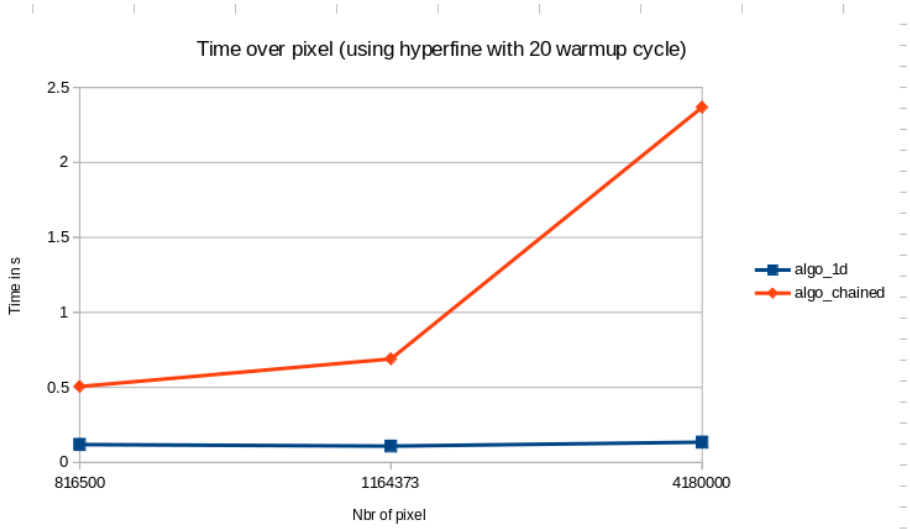
Temps réel selon commande time :



Temps moyen selon hyperfine :



Temps moyen selon hyperfine avec warmup :



Conclusion

On remarque globalement que l'approche liste chainé prend beaucoup plus de temps que l'approche vecteur 1D. Ceci s'explique facilement étant donné que pour accéder à un pixel en particulier nous devons itérer sur les précédents avant de pouvoir l'atteindre. Même dans notre implémentation il reste encore beaucoup d'itération supplémentaire par rapport à la version 1D.

On remarque aussi qu'avec le warmup on obtient des temps plus élevés. Le warmup va permettre de remplir la cache => plus de miss => donc performance moins bonne. Cependant cela nous donne une information des performance de notre code en worst case => plus interessant.