Mohamed Nefsi 300305042

Similarity image search

Partie Concurrente (Go)

Machine Utilise :

Processor Intel(R) Core (TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz Installed RAM 16.0 GB (15.7 GB usable) System type 64-bit operating system, x64-based

Commandline :

go run similaritySearch.go "C:\Users\Ordinateur\Documents\computer_science\CSI2520\ gosimilaritySearch \queryImages\q0 0.jpg" "C:\Users\Ordinateur\Documents\computer_science\CSI2520\gosimilaritySearch\imageDataset3"

RÉSULTATS OBTENUS : Exemple

1. Top 5 similar images to q00.jpg:

Résultats du programme java :

2462.jpg.txt - Similarity: 0.6491030092592595
3756.jpg.txt - Similarity: 0.6620254629629632 3714.jpg.txt
- Similarity: 0.668431712962963
3806.jpg.txt - Similarity: 0.7074537037037038
1144.jpg.txt - Similarity: 1.0

Résultats du programme Go :

1144.jpg - Similarity: 1.0000002
3806.jpg - Similarity: 0.7040053
3756.jpg - Similarity: 0.6606084
3714.jpg - Similarity: 0.6596875
3668.jpg - Similarity: 0.643304

2. Top 5 similar images to queryImages/q01.jpg:

Résultats du programme java :

2592.jpg.txt - Similarity: 0.5049710648148149
1875.jpg.txt - Similarity: 0.5282986111111114
3553.jpg.txt - Similarity: 0.5365624999999999 2536.jpg.txt
- Similarity: 0.561255787037037
3588.jpg.txt - Similarity: 0.5703645833333333

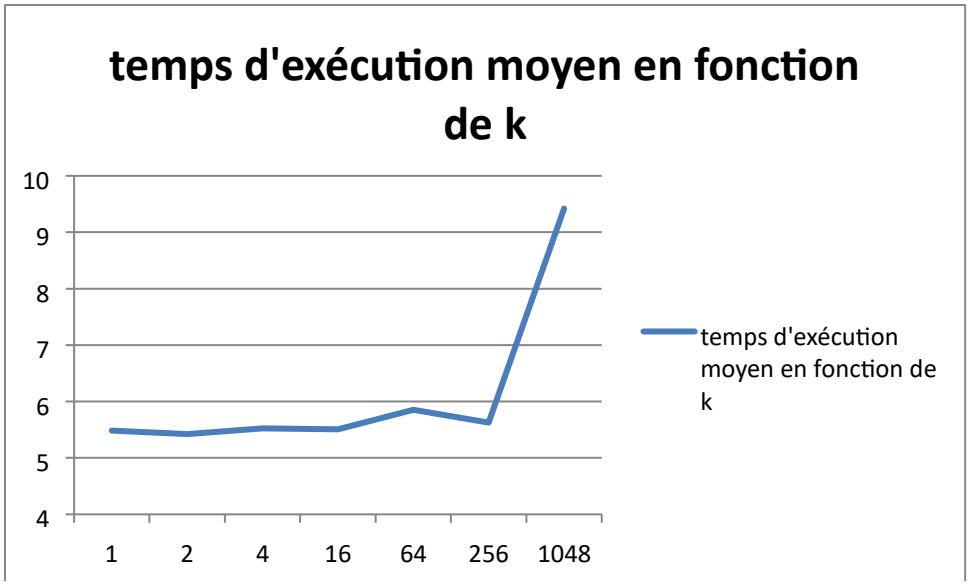
Résultats du programme Go :

3588.jpg - Similarity: 0.572014
2536.jpg - Similarity: 0.560521
3553.jpg - Similarity: 0.536244
1875.jpg - Similarity: 0.528912
2592.jpg - Similarity: 0.50403

Expériences selon la valeur de k :

Résultats :

k	temps d'exécution en s				
	q00	q01	q02	q03	moyenne
1	5,4218339	5,6686651	5,4635895	5,3847573	5,48471145
2	5,4034402	5,4133635	5,5212322	5,3525633	5,4226498
4	5,5772654	5,6904233	5,4554872	5,3693655	5,52313535
16	5,4971239	5,5073499	5,4885602	5,5468462	5,50997005
64	6,7729834	5,6133911	5,5042007	5,5177435	5,85207968
256	5,7120939	5,5935115	5,5622848	5,6300096	5,62447495
1048	9,5967598	9,3922845	9,320753	9,3728716	9,42066723



Interpretation :

1. **Faible valeur initiale avec k = 1** : Quand le travail est effectué par un seul thread, le temps d'exécution est généralement réduit. Ceci s'explique par le fait que la répartition des données à traiter sur un nombre moindre de threads diminue les dépenses liées à la synchronisation et à la gestion des multiples threads.
2. **2 Lorsque le nombre de threads passe de 1 à 4, avec k = 2 et k = 4,** Lorsque le nombre de threads passe de 1 à 4, avec k = 2 et k = 4, on observe une légère réduction du temps d'exécution moyen. Cela suggère une meilleure distribution des tâches, ce qui favorise une accélération du traitement grâce au parallélisme.
3. **Stabilité et légère augmentation avec k = 16 et k = 64** : Après k = 4, le temps d'exécution moyen reste relativement stable jusqu'à k = 64, avec une légère augmentation observée. Cela suggère que le système bénéficie de la parallélisation jusqu'à un certain point, mais au-delà d'un certain nombre de threads (64 dans ce cas), les avantages de la parallélisation commencent à se stabiliser ou même à diminuer en raison des coûts supplémentaires associés à la gestion de plusieurs threads.
4. **Augmentation significative avec k = 1048** : À k = 1048, le temps d'exécution moyen augmente de manière significative.

Cela peut être dû à une surcharge excessive du système avec trop de threads en cours d'exécution, entraînant des conflits de ressources et un temps de latence accru.

En résumé, le graphique montre que le temps d'exécution moyen tend à diminuer jusqu'à un certain nombre de fils (entre 4 et 64 dans ce cas), après quoi il se stabilise ou même augmente légèrement. Cela suggère qu'il existe un équilibre à trouver entre la parallélisation pour améliorer les performances et les coûts supplémentaires associés à la gestion de plusieurs threads.