LEPL1503 - Groupe V2
Code efficace multithreadé en C
Rapport de projet

BOTTON Florentin
florentin.botton@student.uclouvain.be
3008-19-00

D'INTINO SALGADO Lorenzo
lorenzo.dintino@student.uclouvain.be
8028-19-00

ACCOU Pierre
pierre.accou@student.uclouvain.be
0129-18-00

DE RO Corentin
corentin.dero@student.uclouvain.be
1310-18-00

DOERAENE Anthony
anthony.doeraene@student.uclouvain.be
0302-20-00

RIXEN Thomas
thomas.rixen@student.uclouvain.be
1463-20-00

***Résumé*—Implémentation d'un code C afin d'améliorer les performances globales du programme et sa vitesse d'exécution grâce aux threads**

I. MÉTRIQUES DE COMPARAISON DE PERFORMANCE

A. Vitesse d'exécution

L'exécution du code C en multi et single threads nous a permis d'observer une différence de vitesse d'exécution impressionnante. En effet, pour une exécution avec 200 fichiers donnés en input, le temps d'exécution du programme en single-thread est de **2549.5ms**. Avec le même nombre de fichiers en input, mais cette fois en exécutant le programme en multi-threads (4 threads) nous avons obtenu un temps d'exécution de **814.67ms**. Soit plus de 3 fois plus rapide. Quant à la comparaison avec le programme python, après avoir fait des tests de vitesse pour 5, 10, 20, 50 fichiers respectivement, nous pouvons dire que notre programme avec 4 threads est, à peu près, 2400 fois plus rapide que son équivalent python.

B. Mémoire utilisée

Nous avons mesuré la mémoire utilisée par le programme python grâce au package memory-profiler [1] qui nous a permis de générer les résultats utilisés sur le graphique suivant. Nous avons ensuite utilisé valgrind pour mesurer la mémoire utilisée en C. Nous pouvons directement observer une nette amélioration quant à la quantité de mémoire utilisée par notre programme. En effet, seulement à peu près 4.2% de la mémoire utilisée par le programme python est utilisée lors de l'exécution de notre code étant donné que le programme python consomme 32Mo de mémoire et notre programme seulement 1.35Mo.

RÉFÉRENCES

[1] Memory profiler : package python d'analyse de mémoire utilisée (https://pypi.org/project/memory-profiler/), par Fabian Pedregosa and Philippe Gervais, inspiré du travail de Robert Kern