UCLouvain

UCLouvain - Université catholique de Louvain

LINFO1252 : Systèmes Informatiques - P0

Groupe 30 (vendredi)
Abdullahu Eduart : 03132100
Antúnez García Isaías : 03142100

10 décembre 2023

1 Introduction

L'objectif de ce projet est d'apprendre à évaluer et expliquer la performance d'applications utilisant plusieurs threads et des primitives de synchronisation : mutex, sémaphores (première partie) ainsi que les verrous avec attente active, que nous avons implémenté nous mêmes (deuxième partie).

2 Mise en situation

Les applications considérées sont celles vues en cours et en TD : philosophes, producteurs/consommateurs avec un buffer de taille fixe, et lecteurs/écrivains avec la priorité aux écrivains. La métrique d'évaluation que nous considèrerons sera le temps d'exécution total, en variant le nombre de threads d'exécution, et pour un nombre d'opérations fixe (ne dépendant pas du nombre de threads)

3 Philosophes

Nous avons fait face au problème des philosophes présenté au cours et selon les spécifications de l'énoncé du projet. Chaque philosophe effectue 1 million de cycles penser/manger au lieu de 10 millions comme c'était décrit dans les consignes pour avoir un temps plus adéquat (conseil d'un tuteur). Nous pouvons dans chaque programme avoir le mode verbose avec l'argument -v et le lancer avec notre propre implémentation des mutex et semaphores avec -m en ligne de commande quand on lance le programme, ici on va simplement exécuter l'algorithme du problème avec les mutex POSIX et les mutex que nous avons implémentés nous-mêmes par attente active en utilisant des instructions atomiques. Nous recevons le graphe suivant :

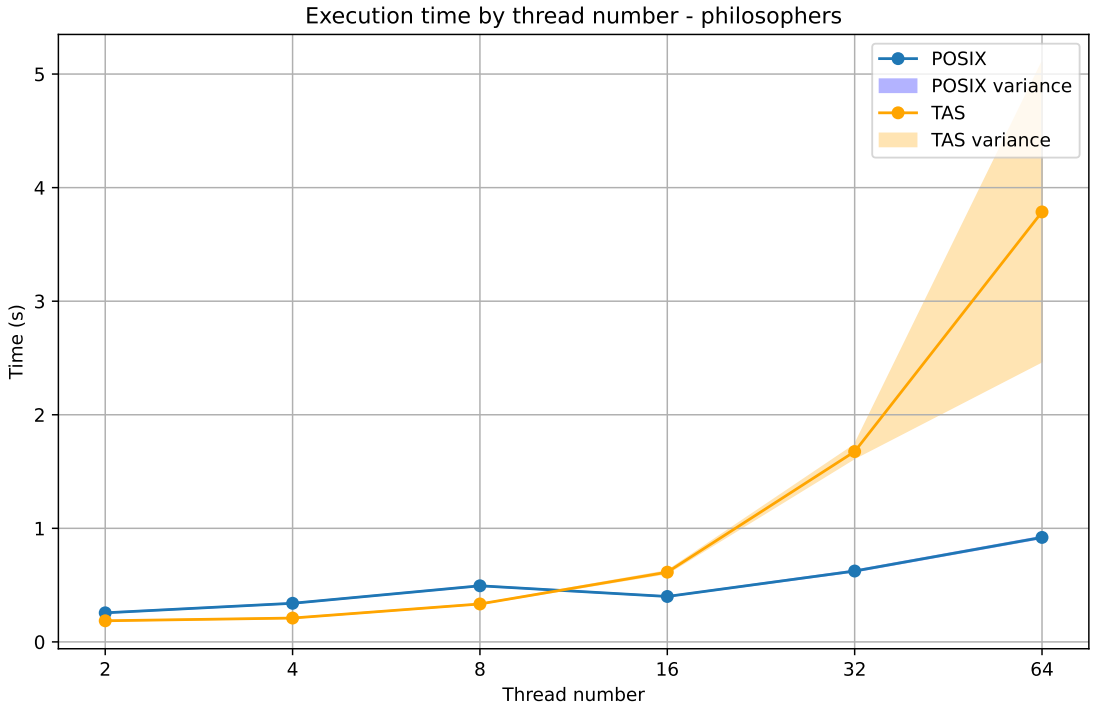

FIGURE 1 – Évaluation des performances du programme philosophes

On peut constater que la courbe du test-and-set monte plus rapidement que celle des mutex POSIX et la variance est beaucoup plus importante. Cependant, nous pouvons voir que le temps d'exécution augmente avec le nombre de threads puisqu'ici le travail augmente avec le nombre de threads et il n'y a pas de partage de travail.

4 Producteur/Consommateur

Nous arrivons au problème des producteurs-consommateurs où le nombre d'éléments produits et consommés est fixe, les threads ne peuvent pas dépasser cette valeur (8192). Au début, ceci posait problème vu que des fois un élément de plus ou en moins était produit/consommé mais grâce au travail en équipe nous avons pu résoudre ce problème. Vu que la différence de performance est importante dans les deux cas, nous avons décidé de séparer le graphe en deux :

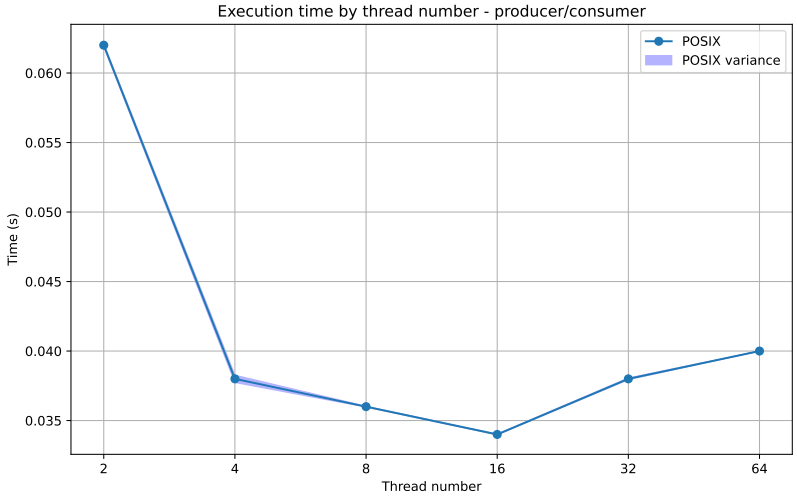

FIGURE 2 – Évaluation des performances du programme producteur/consommateur

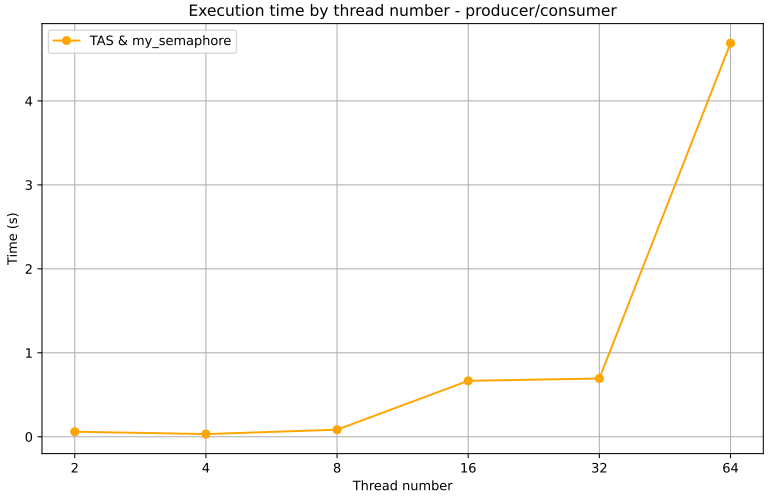

FIGURE 3 – Évaluation des performances du programme producteur/consommateur avec nos implémentations

On commence à constater non seulement une plus grande différence de performance mais aussi une différence de courbature dans les graphes. On est d'avis que notre implémentation des semaphores n'est pas la meilleure vu qu'elle est assez complexe à reproduire. De l'autre côté, on voit bien dans les threads POSIX que la tâche devient plus faisable avec plus de threads qui se partagent la tâche en numéro de cycles que chacun doit faire.

5 Lecteur/Écrivain

Nous arrivons en fin au dernier problème à programmer qui est celui des lecteurs et écrivains. Ici on constate davantage la différence de temps que le programme prend avec plus de threads, ici les écrivains effectuent 640 écritures et les lecteurs 2560 lectures. Ceci devient un travail plus facile avec quand le nombre de threads croît. Malheureusement, nous voyons aussi les résultats de notre implémentation où le temps d'exécution augmente radicalement avec le nombre de threads, surtout de 32 à 64.

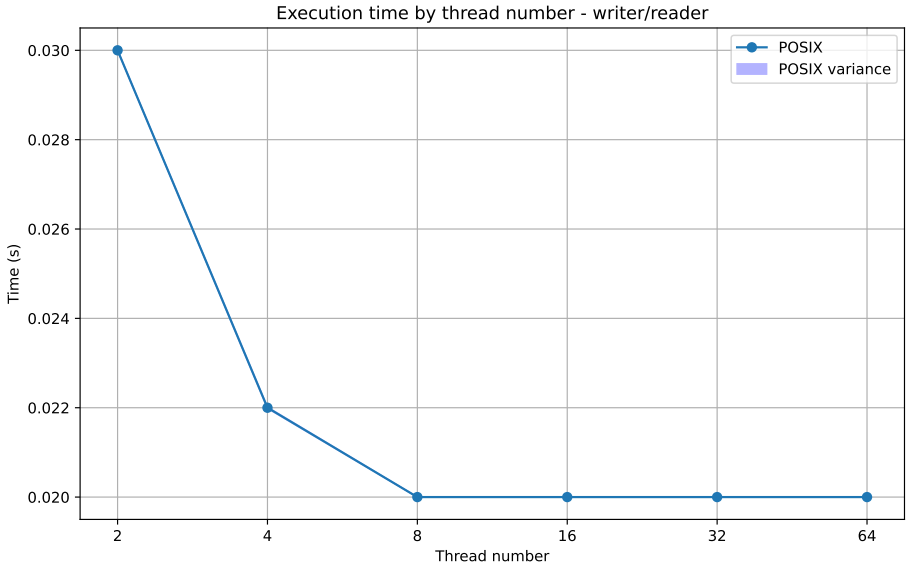

FIGURE 4 – Évaluation des performances du programme lecteurs-écrivainss

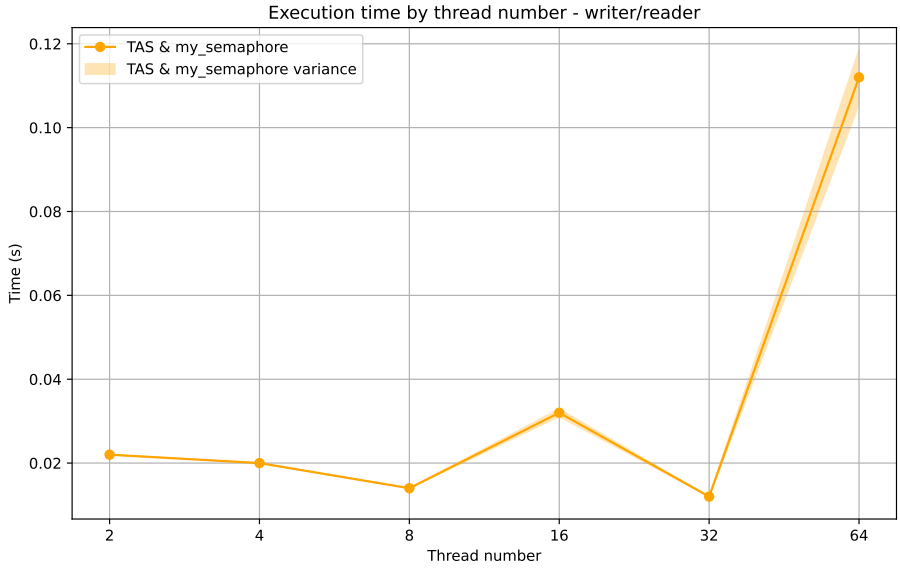

FIGURE 5 – Évaluation des performances du programme lecteurs-écrivains avec nos implémentations

6 Conclusion

Nous pouvons conclure que ce projet nous a permis de comprendre plus en détail les subtilités des threads et les structures comme les mutex et semaphores. Nous sommes bien contents avec notre travail en général sauf avec notre implémentation des semaphores qui a résulté en une difficulté supérieure. Cependant, notre projet fonctionne convenablement et selon les instructions données dans l'énoncé. Ce projet nous a donné l'opportunité de perfectionner notre utilisation du langage c tout en apprenant des fautes et approfondissant dans la matière du cours.