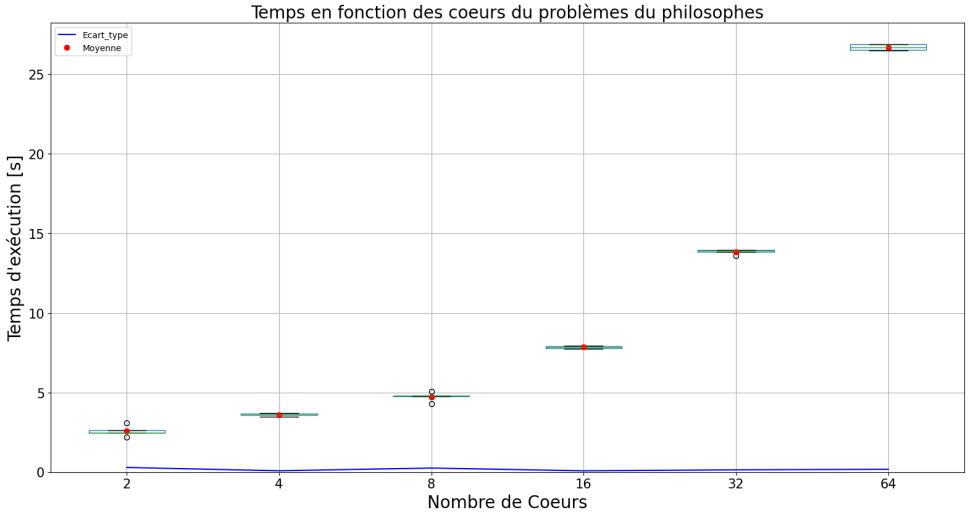
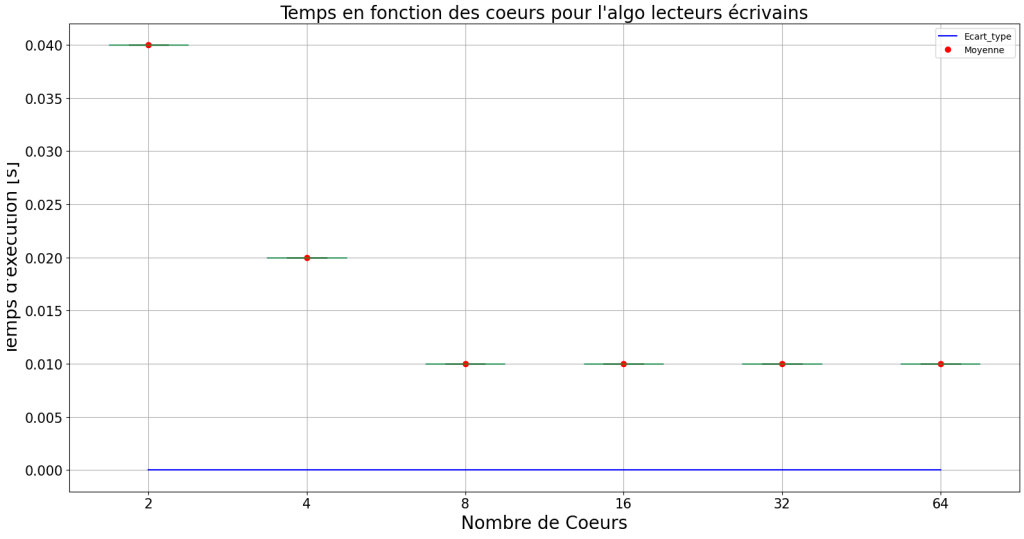
Groupe 40: FORET Felix - 87352100, GALLOWAY Joanna - 03292000

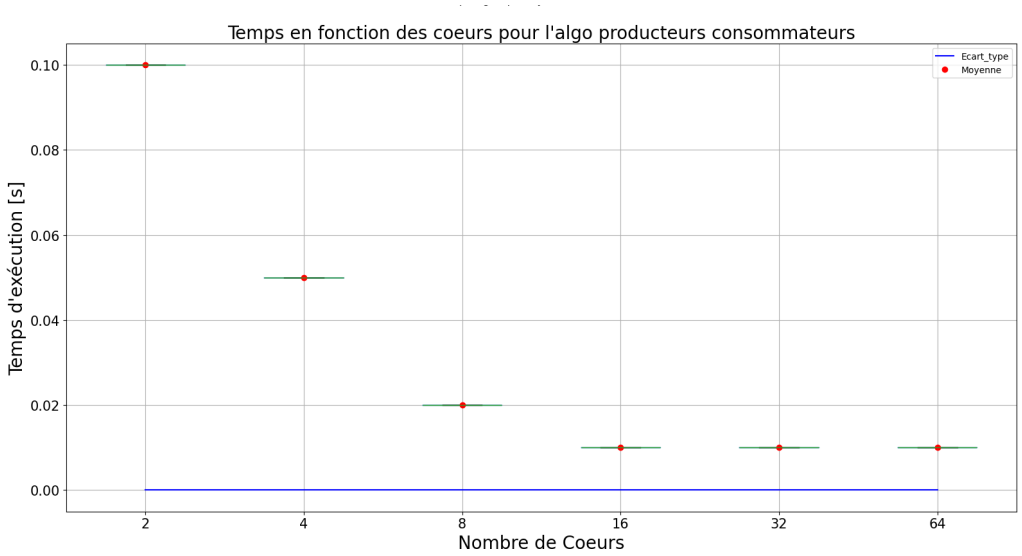
1 Partie 1 - Graph + Analyse



Nous pouvons observer une très faible variation de l'écart type ce qui est une bonne chose. Lors des 5 essais distincts, nous obtenons des mesures quasi identique La performance est consistante. Nous observons également un phénomène étrange, lorsque nous augmentons le nombre de coeur pour ce problème, nous obtenons un plus grand temps d'exécution. Ceci est probablement du au plus grand nombre de philosophes qui veulent accéder à la ressources partagé (les baguettes).



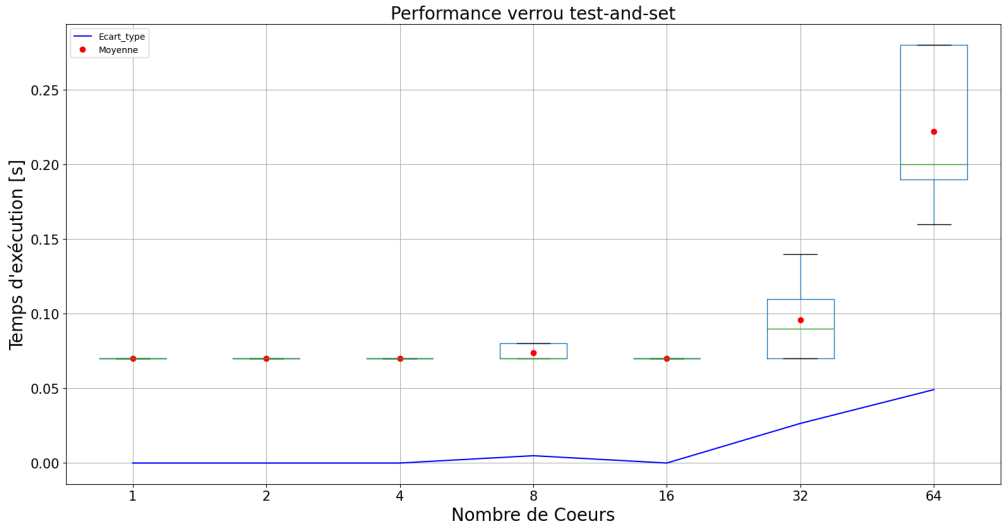
Pour l'exécution du code des lecteurs et des écrivains, nous pouvons observé une diminution du temps d'exécution pour la configuration à 2,4 et 8 coeurs mais au-delà des ces valeurs, le temps semble atteindre un plateau a environ 0.010 secondes. De plus, il y a également aucun écart type donc nous observons les mêmes résultats pour les 5 essais différents.



Pour l'exécution du code des producteurs et des consommateurs, nous pouvons observer une forte diminution du temps d'exécutions entre 2,4 et 8 coeurs ensuite nous atteignons un plateau d'environ 0.01s à partir de 16 coeurs. Nous observons également aucun écart type donc nos temps d'exécutions sont égaux pour les 5 essais différents.

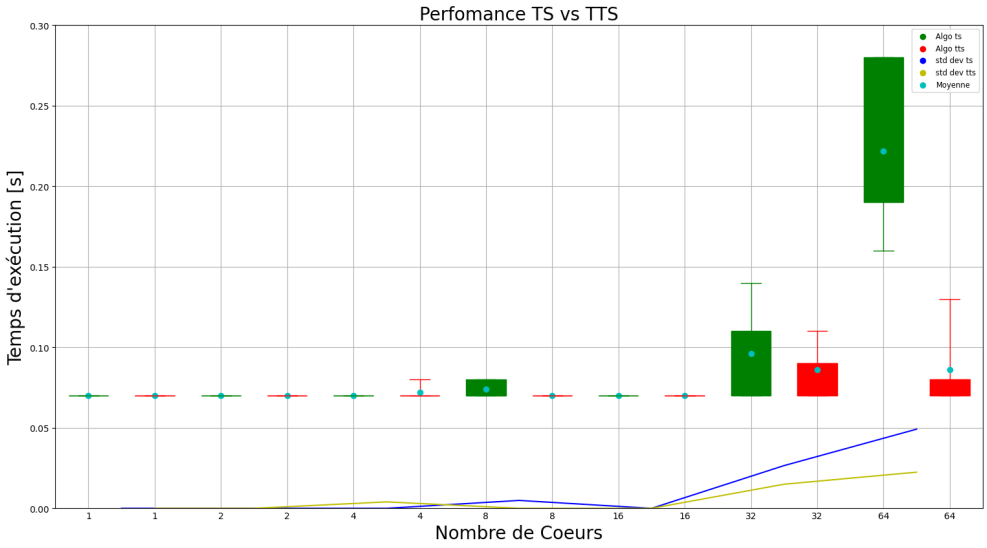
2 Partie 2 - Graphe + Analyse

2.1 Performance du verrou test-and-set



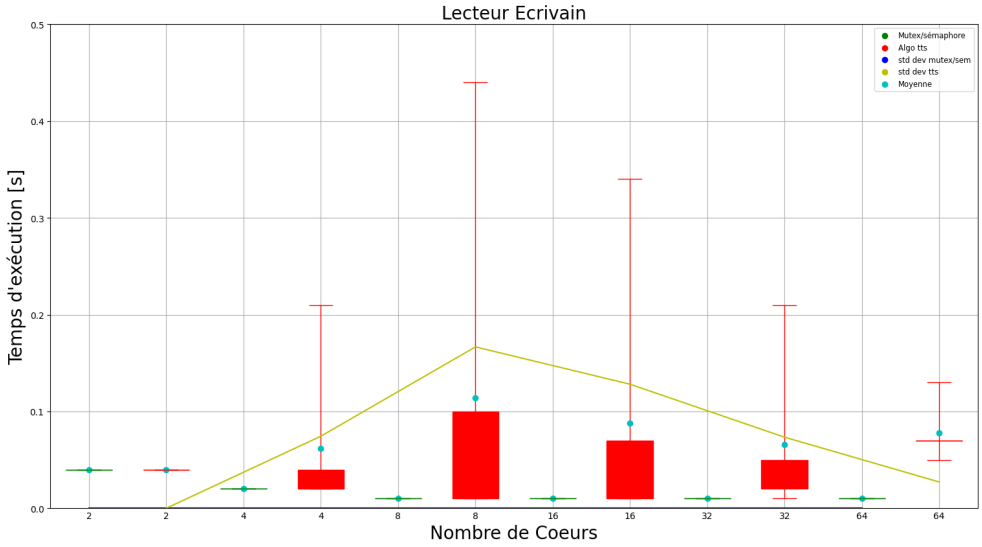
Nous pouvons observer que pour un nombre limité de coeur(1,2 et 4), aucune augmentation du temps d'exécution n'est observé et aucun écart type n'est relevé. Une légère variation est observable autour de 8 coeurs tandis que nous pouvons observer une forte augmentation de temps d'exécution pour 32 et 64 coeurs. Plus nous ajoutons de coeurs, plus nous avons de variations dans nos résultats cependant, la tendance générale est l'augmentation du temps d'exécution.

2.2 Performance du verrou test-and-test-and-set

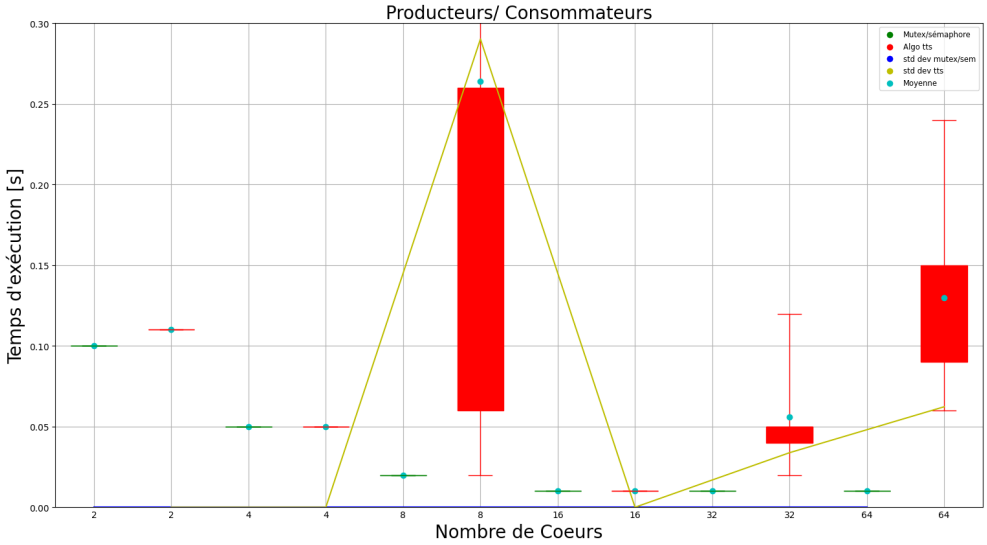


Lorsque nous évaluons la performance de notre Test-and-Set et celle de Test-and-Test-and-Set, il est immédiatement apparent que l'algorithme test and set est le moins performant. Bien que les deux algorithmes ai une plus grande variance au-delà de 32 coeurs, il est notable que l'algorithme Test-and-Set a une performance moins favorable à 64 coeurs que l'algorithme Test-and-Test-and-Test. Cet écart est mise en avant par l'écart type, qui est plus élévé pour 'TS' que pour 'TTS'. Nous ne sommes donc pas garantis de la même performance pour un même problème avec une nombre spécifique de coeur pour l'algorithme 'TS'. L'algorithme 'TTS' est un algorithme plus fiable et plus performant, c'est pour cela que nous l'avons choisis pour tester sur les 3 problèmes testés ci-dessus.

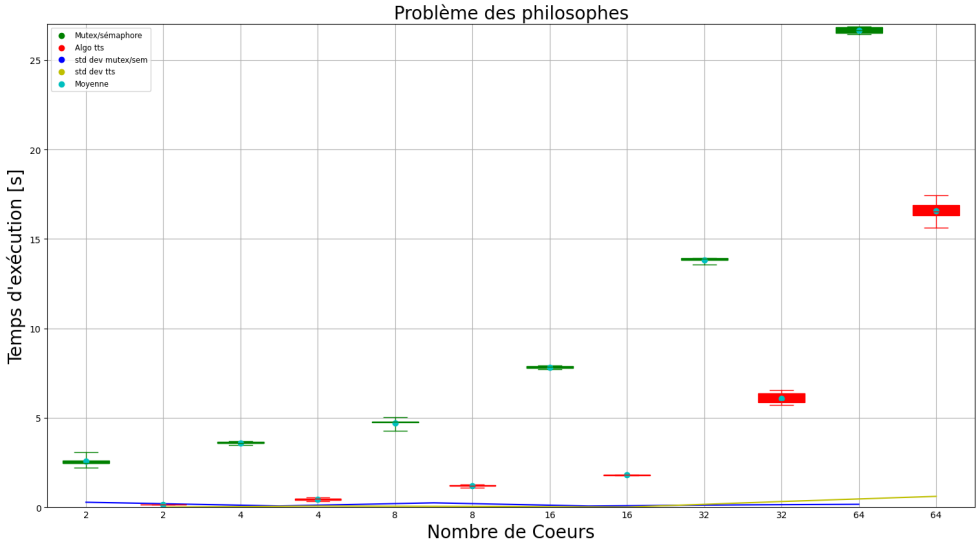
2.3 Comparaison entre TATAS et pthread_mutex pour les trois problèmes précédents



Pour l'applications des lecteurs/écrivains, notre implémentation Test-and-Test-and-Set présente une performance inférieure à celle des mutex et sémaphores déjà implementé en c. A partie de 4 coeurs, la moyenne du TTS est plus élévée avec un temps d'exécution atteignant un pic à 8 coeurs. En contraste, les mutex et des sémaphores déjà implementé ont un écart type nul tandis que l'algorithme TTS présente une forte variance



Pour l'application des producteurs/consommateurs, notre implémentation Test-and-Test-and-Set présente également une performance inférieure à celle des mutex et sémaphores déjà implementé en c. Bien que le temps d'exécution décroît en général en suivant la tendance des mutex/sémaphores, nous remarquons un temps d'exécution extrêmement élevé à 8 coeurs et à 64 coeurs. La moyenne et l'écart type sont nettement plus élevés.



Pour l'application du problèmes des philosophes, nous remarquons que notre implémentation Test-and-Test-and-Set suit la même tendance que celle des mutex et sémaphores déjà implementé en c. Cependant, notre implémentation prenait un temps excessif pour run par conséquent, nous avons limité la boucle 1000000 au lieu de 10000000.Bien que l'écart type de

notre algorithme soit satisfaisant, son temps d'exécution est désastreux pour le problème des philosophes.

3 Conclusion

Notre algorithme présente en générale un écart type élevé ce qui compromet sa capacité à produire des résultats cohérents.Bien que notre performance soit acceptable pour les problèmes des producteurs/consommateurs et des lecteurs/écrivains, elle demeure moins efficace. Malheureusement, pour le problème des philosophes, notre algorithme ne fournit pas des résultats satisfaisants.