Description du travail réalisé: Parallélisation avec OpenMP & MPI
AMS I-03

Jincheng KE

21 janvier 2024

1 Introduction

Ce rapport présente le travail réalisé dans le cadre du projet de parallélisation d'un code de calcul scientifique en utilisant OpenMP et MPI. L'objectif était d'optimiser les performances sur un processeur AMD R9 3900 sous Windows 11, en comparant différentes approches de parallélisation.

2 Environnement de Développement

Le projet a été développé sur un système d'exploitation Windows 11. Pour la compilation du code, le compilateur Intel icpx de l'Intel oneAPI a été choisi pour sa compatibilité et son efficacité, bien qu'il ne soit pas entièrement optimisé pour les processeurs AMD comme celui utilisé (AMD R9 3900, 12 cœurs, 24 threads).

Cette fois, nous avons utilisé une approche de programmation hybride en combinant OpenMP et MPI pour explorer l'impact de cette combinaison sur les performances. L'objectif était de comprendre comment l'intégration de ces deux stratégies pourrait influencer l'efficacité du traitement parallèle.

3 Stratégie Expérimentale

Dans notre projet, nous avons adopté une stratégie de parallélisation hybride en combinant MPI et OpenMP. Le principe était de diviser l'espace de calcul en plusieurs sous-espaces gérés par différents processus MPI, et au sein de chaque sous-espace, d'effectuer des calculs en parallèle à l'aide de threads OpenMP. Cette approche a été mise à l'épreuve à travers une série d'expérimentations impliquant différentes configurations de processus et de threads, telles que 16 processus avec 1 thread, 8 processus avec 2 threads, 4 processus avec 4 threads, 2 processus avec 8 threads et 1 processus avec 16 threads. L'objectif était de comparer les performances

de chaque configuration en mesurant le speedup obtenu, afin de déterminer quelle combinaison offrait la meilleure efficacité en termes de traitement parallèle.

4 Résultats Expérimentaux

Les résultats des expériences montrent des performances variables en fonction de la combinaison des processus MPI et des threads OpenMP. Les configurations testées comprenaient 16 processus avec 1 thread, 8 processus avec 2 threads, 4 processus avec 4 threads, 2 processus avec 8 threads et 1 processus avec 16 threads. Les configurations testées et leurs speedups respectifs sont les suivants :

— 1 processus avec 16 threads : Speedup de 7.35.
— 16 processus avec 1 thread : Speedup de 4.94.
— 2 processus avec 8 threads : Speedup de 7.67.
— 4 processus avec 4 threads : Speedup de 7.71, le meilleur résultat.
— 8 processus avec 2 threads : Speedup de 6.78.

Ces résultats indiquent que la performance est influencée par l'équilibre entre le nombre de processus MPI et de threads OpenMP. Avec un nombre élevé de processus, la communication entre les processus MPI devient un facteur limitant, réduisant ainsi les performances. En revanche, la configuration de 4 processus et 4 threads offre le meilleur équilibre entre la parallélisation fine et grossière, optimisant la gestion de la charge de travail et les coûts de communication. Dans des sous-espaces plus restreints, la charge par thread est plus uniforme et les coûts de communication sont relativement limités, ce qui mène à de meilleures performances globales.

5 Conclusion

Ce projet a illustré les avantages et les défis de l'utilisation combinée de MPI et OpenMP pour la parallélisation. L'expérimentation a montré que l'équilibre optimal entre la parallélisation grossière (MPI) et fine (OpenMP) dépend fortement de la nature de la tâche et de l'environnement de calcul. Alors que l'utilisation de nombreux processus MPI augmente la communication inter-processus, réduisant l'efficacité, une approche hybride équilibrée, comme dans la configuration de 4 processus avec 4 threads, a permis d'atteindre les meilleures performances en minimisant la communication tout en maximisant le parallélisme. Ce projet souligne l'importance d'une planification et d'une optimisation soigneuses dans la conception de solutions parallèles, spécialement dans des environnements complexes où plusieurs types de parallélisation sont possibles.