Etude de code de parallélisation en langages de programmation c et c++

Fiche d'expérimentation et d'analyse

Supervisé par :
Dr. MESSI

Faculté des Sciences

Nom :	**DJIEMBOU TIENTCHEU VICTOR NICO**
Devoir :	2
Lien :	drive.google.com/drive/folders/1YMUePascSgc4nJK2Cw4BFDG6IaDJ0Eap
Reçu le :	2023.11.29
Auteur :	Perçu de Dr. Messi
Mots clés :	Distributed and parallel ML, c, c++, threads
UE :	INF5099
Matricule :	17T2051
Département :	Informatique
Niveau - Option :	M2 - Sciences des Données

Table des matières

1 Contexte

À l'université de Yaoundé 1, en classe de Master 2 du cycle de Master les étudiants sont appelés à s'initier à la recherche et à monter en compétence dans des domaines fare des sciences informatiques.

En option sciences des données, nous avons parmi autre pour objectif de pouvoir améliorer les performance de nos processus d'apprentissage. C'est dans cette optique que nous nous focalisation dans la distribution de nos apprentissage en particulier dans des langages de développement c et c++.

L'enseignant pour nous plonger dans ce decor a donc jugé nécessaire de nous faire visiter quelques cas de base de code de parallélisation.

2 Problème

Etant appeler à proposer des solution d'apprentissage distribués en langages c et c++, le problème est de savoir comment implementer un code parallélisé ? Quels sont les outils qui dans ses langages serve la cause de la parallélisation de processus ?

3 motivations

Il serait intéressant pour nous étudiants de pouvoir à partir de cours de pouvoir expertiser sur

— comment penser et mettre sur pieds un projet avec logique de parallélisation dans les langages c et c++ ?
— quels sont les librairies utilisées dans ses distincts langages pour paralléliser le code ?
— quels sont les contraintes d'implementation dans chaque langage ?

4 Parallélisation dans le langage c++

Sur la base des codes **code_parallel_cpp** et **produit_scalaire_cpp**, nous pouvons avoir un début de compréhension sur comment se passe la parallélisation sous le langage c++.

Déjà, à la lumière de mon analyse, j'ai décélé 2 grands niveau de fourniture de la parallélisation. Déjà nous devons pouvoir `mettre sur pieds un code fonctionnel à logique parallèle` puis nous devons `pouvoir fournir les ressources nécessaire lors de la compilation du fichier binaire d'exécution`. À la Figure 1, nous présentons avec commentaire à l'appui comment est-ce que une code c++ peut être parallélisé moyennant la bibliothèque **thread** qui permet de programmer des solutions qui utilisent plusieurs threads. Ensuite, le code étant conçu de tel sorte à autoriser une exécution parallèle, il faudrait bien encore que lors de la compilation, nous explicitions à compilateur GCC de c++ (g++) qu'il s'agit d'un code qui utilisera une logique mutlithread (voir Figure 2).

5 Parallélisation dans le langage c

L'ajout d'une logique de manipulation multithreading en c nécessite impérativement l'inclusion dans le projet de la bibliothèque **pthread.h**. De plus contrairement à c++, c est plus bas niveau, donc il faut tenir compte de bien plus d'aspect durant la parallélisation tels que le **verrouillage**, **déverrouillage**, **lancement et arrêt d'un thread** des ressources avec un service de **mutex** qu'offre la bibliothèque tel que **pthread_mutex_init, pthread_mutex_lock, pthread_mutex_unlock, pthread_attr_setdetachstate, pthread_exit** (voir plus de détails dans les Figures 3, 4, 5). En ce qui concerne le Makefile de compilation et d'exécution, il demeurre identique à celui de c++ à la seul différence du compilateur GCC de c qui est gcc au lieu de g++.

6 Conclusion

La programmation parallèle est un concept très répandu dans le domaine d'apprentissage automatique et est accessible dans deux nombreux langages tels que c et c++. Dans ses derniers, il est toujours important de codé avec les outils mise en oeuvre pour la programmation mutlithread tels **thread** et **pthread.h** respectivement en c++ et c, et d'associer lors de la compilation de la version binaire, les ressources permettant cette exécution parallèle grâce à des flag qui on été préalablement mise en place dans le compilateur gcc qui sont entre autres **-lpthread -fopenmp**.

Références

[Bre09] Clay Breshears, *The art of concurrency : A thread monkey's guide to writing parallel applications*, O'Reilly Media, 2009.

[KKI03] George Em Karniadakis and Robert M. Kirby II, *Parallel scientific computing in c++ and mpi : A seamless approach to parallel algorithms and their implementation*, Cambridge University Press, 2003.

[Pac11] Peter Pacheco, *Parallel programming in c++ and mpi*, Morgan Kaufmann, 2011.

[Qui03] Michael J. Quinn, *Parallel programming in c with mpi and openmp*, McGraw-Hill Education, 2003.

[WA04] Barry Wilkinson and Michael Allen, *Parallel programming : Techniques and applications using networked workstations and parallel computers*, 2nd ed., Prentice Hall, 2004.

[Wil12] Anthony Williams, *C++ concurrency in action : Practical multithreading*, Manning Publications, 2012.

[Wil19] ———, *C++ concurrency in action : Practical multithreading*, Manning Publications, 2019.

7 Annexe

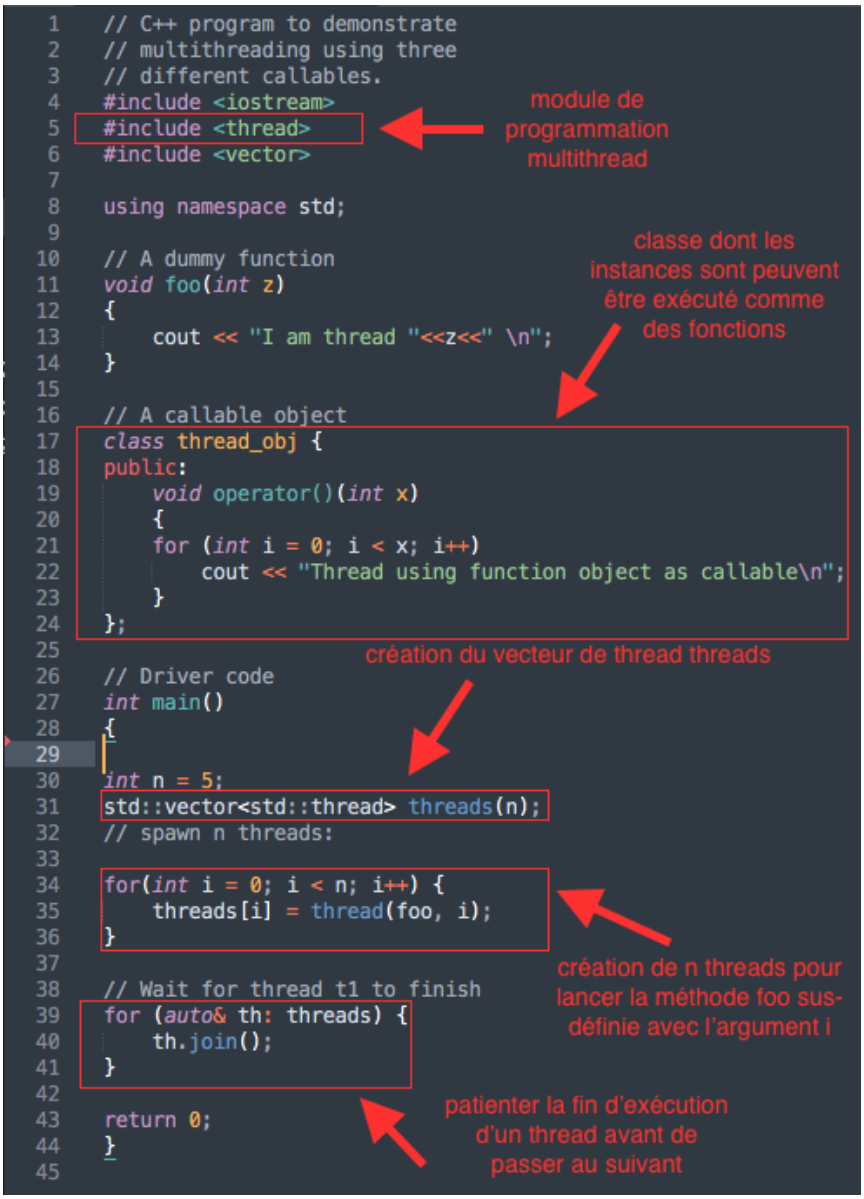

FIGURE 1: Explication des parties du code parallèle

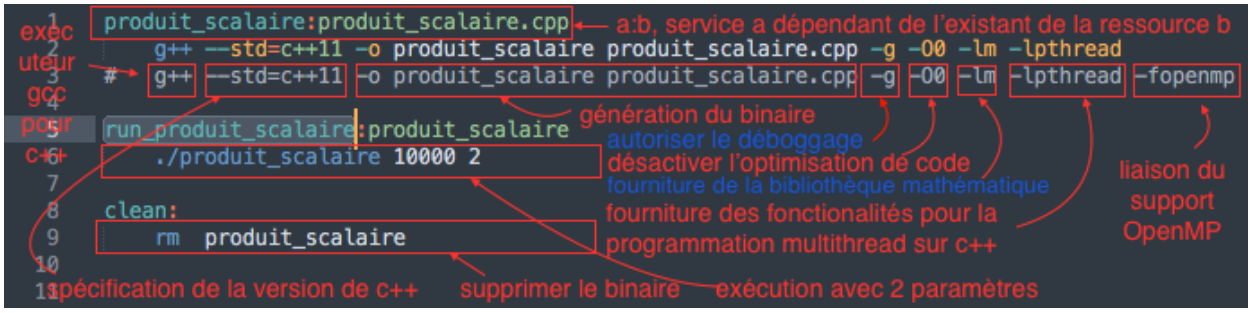

FIGURE 2: Explication des ressources nécessaire pour la compilation d'un code parallèle

FIGURE 3: Importation de la bibliothèque pour la programmation parallèle en c

FIGURE 4: Mise en oeuvre de la programmation parallèle en c

FIGURE 5: Verrou transactionnel entre threads