

Support de MPI/OpenMP et de la vectorisation dans Verificarlo

Master Calcul Haute Performance et Simulation

Hery ANDRIANANTENAINA

Ali LAKBAL

Nicolas BOUTON

Encadrant: Eric PETIT

Année 2020-2021

- Compilateur : **Clang et llvm**
- Domaine d'utilisation : **Instrumentation des opérations flottantes**

1. Vectorisation dans le calcul scientifique

1.Introduction :

- **Compilateur** : *Clang et gcc*
- **probleme** : *Etant donné que notre encadrant nous a dit que le support de gcc était éphémère dû à une dépendance avec fortran*
- **solutions** : *supporter les types vectoriels de clang et non pas ceux de gcc.*
- **test** : *pour tester il faut bien configurer **verificarlo** avec **clang** pour le C et C++ avec la commande suivante :*

`./configure --without-flang CC=clang CXX=clang++`

2.Tests :

- *Suivre le fonctionnement de test que Verificarlo a commencé à implémenter.*
- *Nous avons testé si les résultats obtenus lors de la compilation et de l'exécution sont exactes.*
- *Les tests sont principale- ment écrits en bash, avec un code de test écrits en c et un code python*
- *Les tests se trouvent dans le répertoire `tests/test-vector-instrumentation/`.*

2. Tests :

- *Donc en général nous avons effectué des tests sur les opérations arithmétique vectorielles avec les jeux d'instruction sse ,avx et avx512, et s'assurer du bon fonctionnement.*
- *Nous avons testé si les résultats obtenus lors de la compilation et de l'exécution sont exactes.*
- *Nous avons effectué trois sous tests : le bon résultat des opérations vectorielles ,l'appel aux probes vectorielles et l'utilisation des jeux d'instructions vectorielles .*

2.1 :le bon resultat des opérations vectorielles

- *Dans ce cas nous avons testé sur les différents backends, les différentes operation vectorielles avec les vecteurs de taille differente sur les précisions qu'on a choisit (float et double); et on a deduit que les résultats retournés sont vrai*

2.2 :l'appel des probes vectorielles

- *Nous avons généré le fichier intermédiaire pendant la compilation avec la commande : **-save-temps***
- *une fois le fichier généré , on remarque que on a effectivement fait appel à notre probe vectorielle*

2.3 :Utilisation des jeux d'instructions vectorielles

- Dans *verificarle*, les instructions vectorielles pour les opérations arithmétiques sont présentées par la concaténation suivante : *operationvectorielprécision*
- elle s'utilise sur les registres *xmm,ymm,zmm* associés respectivement au jeux d'instructions *sse,avx,avx512* si on prend comme exemple : ***mulps*** avec un registre *xmm* c'est une instruction vectorisée

- *les vecteurs 256 et 512 bits sont déjà inclus et supportés*
- *Dans la premiere version de verifcarlo , les probes vecctorielles sont implémentées mais elles appellent toujours la version scalaire.donc de notre coté on a rajouté des fonction vectorielles dans les backends que nous avons appelé à partir des wrappers dans les probes,dans le fichier `src/vfcwrapper/main.c`*

- *ensuite il faut ajouter les fonctions vectorielles dans l'interface qui se trouve dans le fichier `src/common/interflop.h`*
- *Nous avons cherché et testé comment minimiser le nombre des fonction ,donc la meilleur solution q'uon a trouvé est de mettre la taille des vecteur en paramètre; ca nous a minimisé le nombre à 8 fonctions vectorielles en tout*
- *Comme nous passons la taille en argument, il faudra tester la taille pour permettre à clang d'effectuer une opération vectorielle en changeant le type de notre tableau dans le bon type vectorielles de clang.*
- *De plus nous avons déplacés la définitions des types vectorielles dans le fichier `src/common/inteflop.h` .*

Changements aux niveaux des backends

Fonctions vectorielles en mode scalaire

Tous les backends

Fonctions vectorielles en mode vectoriel

- ieee
- vprec

Changements aux niveaux du backend vprec

Fonctionnement du backend

- norme IEEE754
- fonction de debug

Opérande constantes

- avertissement de clang sur les types des paramètres de fonction
- ajout d'un pragma pour retirer l'avertissement

Fonctionnement du backend

- nombres fini et infini
- nombres normaux et dénormaux



Figure – Représentation d'un nombre flottant simple précision

Ajout dans verifcarlo

Compilation des **wrappers** et des **backends** avec le drapeau
-march=native

Types vectorielles

Vecteur de 4 double précision

Jeu d'instruction disponible

SSE

Clang

Utilise 4 addition vectoriel SSE

Verificarlo

- Backend : vectorisé comme pour clang
- Problème : vecteur passé par registre entre les modules

Cours en relation

Architecture Parallèle