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

Verificarlo

Compilateur de base pour verificarlo

- CLANG
- LLVM

Domaine d'utilisation de verificarlo

Verificarlo permet par instrumentation des opérations flottantes, de pouvoir déboguer les erreurs, dû à la précision machine.

Vectorisation dans le calcul scientifique

Jeux d'instruction

- 128 = sse
- 256 = avx
- 512 = avx512

Compilation

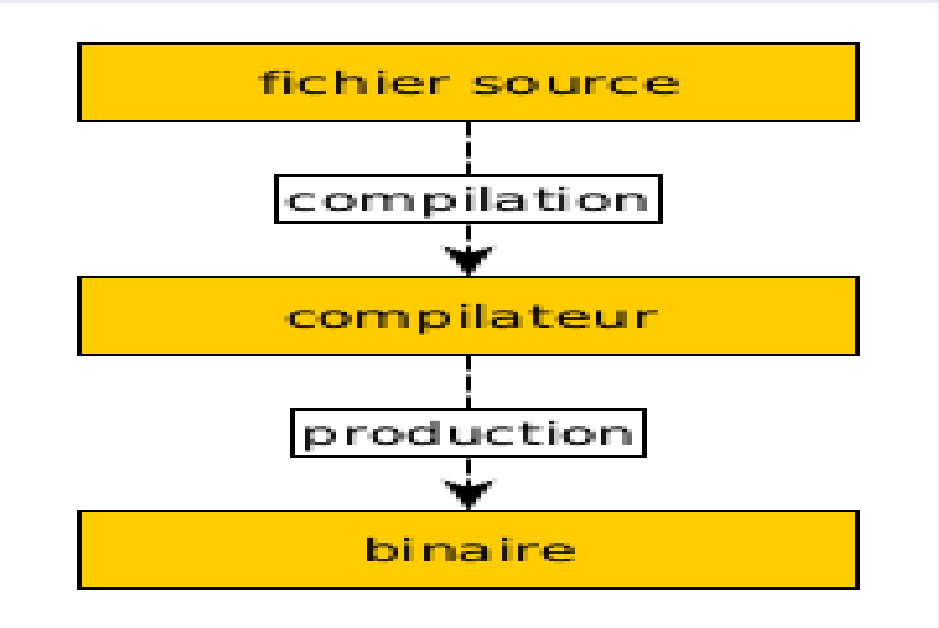

Figure – Fonctionnement de base d'un compilateur

Compilation pour verificarlo

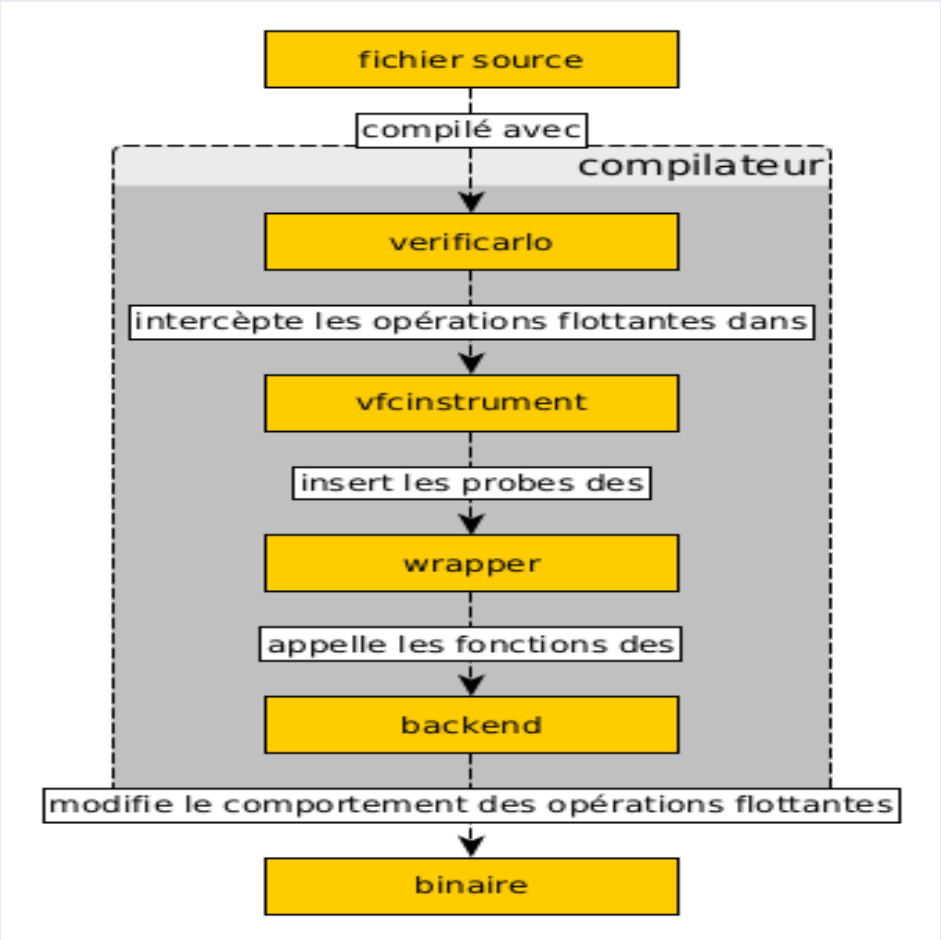

Figure – Fonctionnement de verificarlo

Définition de certains termes techniques

- probes : Les probes sont des fonctions implémenté dans vfcwrapper qui est linker avec le programme par la partie compilation de verificarlo.
- backend : Dans le cadre de verifcarlo, c'est la/les librairie(s) dynamique(s) qui seront appelées par le wrapper dans les probes. Dans le cadre d'un compilateur c'est la derniere phase qui descend de la représentation intermédiaire vers le binaires
- wrapper : Ce sont des fonctions qui enveloppent l'appel à d'autres fonctions.
- link : Il s'agit de la phase de compilation qui consiste à aller chercher toute les librairies externes appelé par l'application pour les liées au programme utilisateur afin de résoudre les références non défini.
- sérialisation : Dans le contexte de l'utilisation de vecteur il s'agit d'exécuter en séquence les éléments du vecteur.

Support MPI/openMP

Notion indispensable pour le parallélisme

- Système à mémoire partage
  - SMP
  - NUMA
- Système à mémoire distribuée
- Thread ou flot d'exécution
- Processus parallèle

Présentation d'open MPI

- Installation : source :
  https ://www.open-mpi.org/software/ompi/v4.1/
- Configuration : ./configure –prefix='/chemin/bin'
- Compilation : **make**
- Installation : **sudo make install**
- Préparation de l'environnement :

Présentation d'open MPI

Description de communication dans Open MPI

- l'environnement d'exécution
- les communication point à point
- les communication collectives
- les groupes de processus
- les topologies de processus

Compilation d'un programme parallèle avec verificarlo

CC=OMPI_CC=verificarlo mpicc

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++
```

Tests

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

Tests

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

sous test 1 / : le bon resultat des opérations vectorielles

- *Dans ce cas nous avons testé sur les differents backends, les differentes 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*

sous test 2 / : l'appel des probes vectorielles

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

sous test3 / :Utilisation des jeux d'instructions vectorielles

- *Dans verificarle, les instructions vectorielles pour les opérations arithmetiques 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*

Vectorisation

support des vecteurs 512/256 bits

- les vecteurs 256 et 512 bits sont déja inclus et supportés

Ajout des probes vectorielles

- *Dans la premieres version de verificarlo , les probes vecctorielles sont implémentées mais elles appelent toujours la version scalaire.donc de notre coté on a rajouté des fonction vectorielles dans les backends que nous avont appelé à partir des wrappers dans les probes,dans le fichier* ***src/vfcwrapper/main.c***

Ajout des fonctions vectorielles dans l'interface

- *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 paramétre ; 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

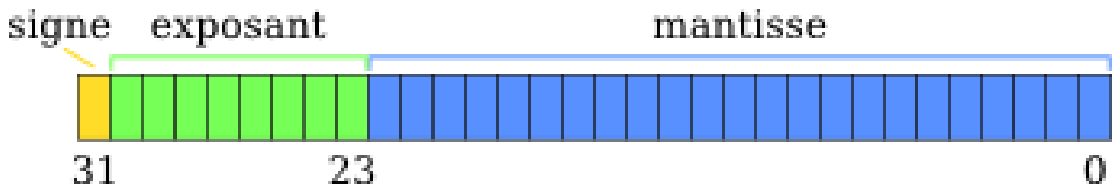

Figure – Représentation d'un nombre flottant simple précision

Compilation

Ajout dans verificarlo

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

Problèmes rencontrés

Types vectorielles

Vecteur de 4 double précision

Jeu d'instruction disponnible

SSE

Clang

Utilise 4 addition vectoriel SSE

Verificarlo

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

Conclusion

Cours en relation

Architecture Parallèle