

# Support de MPI 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 de base pour verifcarlo

- CLANG
- LLVM

## Domaine d'utilisation de verifcarlo

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

- 2\*double = sse
- 4\*double = avx
- 8\*double = avx512

# Définition de certains termes techniques

- 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 toutes les bibliothèques externes appelées par l'application pour les lier au programme utilisateur afin de résoudre les références non définies.
- probes : Les probes sont des fonctions implémentées dans vfcwrapper qui sont liées avec le programme par la phase de compilation de verifcarlo.
- backend : Dans le cadre de verifcarlo, c'est la/les bibliothèque(s) dynamique(s) qui seront appelées par le wrapper dans les probes. Dans le cadre d'un compilateur c'est la dernière phase qui descend de la représentation intermédiaire vers les binaires.
- sérialisation : Dans le contexte de l'utilisation de vecteur il s'agit d'exécuter en séquence les éléments du vecteur.

# 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 verifcarlo

`CC=OMPI_CC=verifcarlo mpicc`

## Bibliography

- <https://www.open-mpi.org>
- <https://fr.wikibooks.org>

## Compilation

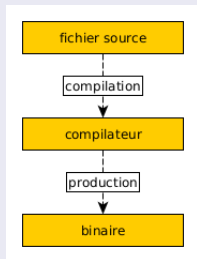


Figure – Fonctionnement de base d'un compilateur

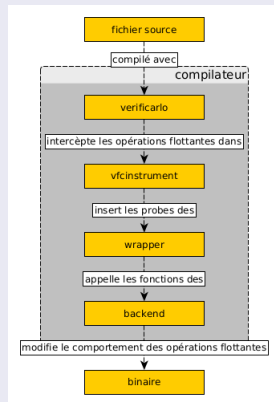


Figure – Fonctionnement de verificarlo

# Changements aux niveaux des backends

## Backend existant

ieee / vprec / mca / bitmask / cancellation / mca-mpfr

## Fonctions vectorielles en mode scalaire

- mode par défaut
- tous les backends

## Fonctions vectorielles en mode vectoriel

- backend ieee
- backend vprec

# Changements aux niveaux du backend ieee

## Fonctionnement du backend

- norme IEEE754
- fonction de débogue

## Opérandes constantes

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

# Changements aux niveaux du backend vprec



Figure – Représentation d'un nombre flottant simple précision

## Cas spéciaux

- NaN (Not a Number)
- nombres dénormaux



## Ajout à la compilation de Verificarlo

Compilation des **wrappers** et des **backends** avec le drapeau :

**-march=native**

## Avantage

Détection automatique des jeux d'instructions disponibles

## Languages

- bash (script de test)
- c (programme principale)
- python

## Test

- opérations arithmétiques vectorielles

## Bon resultat des operations vectorielles

```
float + 4  
2.100000  
2.100000  
2.100000  
2.100000
```

## Verification de l'appel aux probes vectorielles

```
%59 = call <4 x float> @_4xfloatadd(<4 x float> %55, <4 x float> %56)  
...  
%65 = call <4 x float> @_4xfloatmul(<4 x float> %61, <4 x float> %62)  
...  
%71 = call <4 x float> @_4xfloatsub(<4 x float> %67, <4 x float> %68)  
...  
%77 = call <4 x float> @_4xfloatdiv(<4 x float> %73, <4 x float> %74)
```

## Utilisation des jeux d'instructions vectoriels

```
float4
2c24:c5 f8 58 c1      vaddps %xmm1,%xmm0,%xmm0
2c43:c4 c1 78 58 07    vaddps (%r15),%xmm0,%xmm0
Instruction addps and register xmm INSTRUMENTED
3024:c5 f8 59 c1      vmulps %xmm1,%xmm0,%xmm0
3043:c4 c1 78 59 07    vmulps (%r15),%xmm0,%xmm0
Instruction mulps and register xmm INSTRUMENTED
2e24:c5 f8 5c c1      vsubps %xmm1,%xmm0,%xmm0
2e43:c4 c1 78 5c 07    vsubps (%r15),%xmm0,%xmm0
Instruction subps and register xmm INSTRUMENTED
3224:c5 f8 5e c1      vdivps %xmm1,%xmm0,%xmm0
3243:c4 c1 78 5e 07    vdivps (%r15),%xmm0,%xmm0
Instruction divps and register xmm INSTRUMENTED
```

# Conclusion de la Vectorisation

## Fait

- test opérations vectorielles simple
- probes vectorielles
- fonctions vectorielles (mode scalaire ou vectoriel)
- activation des jeux d'instruction

## Reste à faire

- test des conditions vectorielles
- test des opérations vectorielles spécifique aux backends
- vectoriser les backends manquants
- test des performances

# Conclusion

## Problèmes rencontrés

<https://github.com/Safecarlo/verificarlo/issues/25>

## Domaines étudiés

- Parallélisation
- Vectorisation

## Cours en relation

Architecture Parallèle