AGENIUM SCALE

Résultats des benches pour NSIMD sur EFISPEC 3D

15 septembre 2020

À propos de ce document

Ce document présente les résultats de l'exécution d'un noyau de calcul pour la méthode des éléments finis spectraux 3D (EFISPEC3D) sur différentes architectures matérielles. Pour chacun des cas, nous avons utilisé trois versions différentes de l'algorithme : une version scalaire de référence, une version vectorisée à l'aide du jeu SIMD nati de l'architecture cible, et une version utilisant notre bibliothèque de calcul NSIMD.

La bibliothèque NSIMD

Agenium Scale développe et maintient depuis 2019 la bibliothèque Open Source NSIMD, dont la première version est disponible sur Github [1]. Il s'agit d'une bibliothèque C et C++ permettant d'écrire du code SIMD indépendamment d'une architecture particulière. Un code écrit à l'aide de NSIMD peut ainsi s'exécuter sur toutes les architectures supportées par la bibliothèque, ce qui permet de diminuer les coûts de développement liés à la vectorisation d'un algorithme.

NSIMD propose des opérateurs SIMD génériques qui sont, si possible, de simples wrappers vers les opérateurs correspondants sur l'architecture cible. C'est le cas pour les opérations arithmétiques de base, comme l'addition, ou la multiplication par exemple. Dans le cas contraire, les opérateurs sont émulés sur l'architecture, en utilisant de préférence des opérations entre registres SIMD.

NSIMD n'est pas la seule bibliothèque proposant d'abstraire les opérations vectorielles, mais, à notre connaissance, elle est la seule à proposer un support pour le jeu d'instructions SVE dont les premiers processeurs seront bientôt disponibles. NSIMD permet également d'écrire du code SIMD et gère de manière transparente les tailles de registres. Un code écrit à l'aide de NSIMD permet de générer à la fois des exécutables optimisés sur une architecture dont la taille des registres est fixe, ou sur une architecture (comme SVE), où la taille de ces registres n'est connue qu'à l'exécution.

Fonctionnement

La bibliothèque NSIMD repose sur les capacités des compilateurs à *inliner* les fonctions (c'est-à-dire, remplacer leurs appels dans le programme par le code correspondant) lors de l'utilisation de l'option de compilation adéquate (en général, `-O2` et au-delà). De cette manière, un appel à une fonction de NSIMD est directement remplacé par le code ou l'opérateur SIMD correspondant. Cela permet d'abstraire du code SIMD sans coût supplémentaire. Les principaux compilateurs, comme GCC, Clang, ICC et MSVC, proposent cette passe d'optimisation. Afin de permettre ces optimisations, la plupart du code source de NSIMD est placée dans des fichiers d'en-tête, seules les plus grosses fonctions sont placées dans des fichiers sources à part, et compilées sous forme d'une bibliothèque dynamique.

Jeux d'instructions supportés par NSIMD

Actuellement, NSIMD support les jeux d'instructions suivants :

- Intel :
  - SSE2
  - SSE4.2
  - AVX
  - AVX2
  - AVX512 (support pour KNL et Xeon Skylake)
- Arm :
  - NEON 128 bits (ARmv7 et Aarch64)
  - SVE

Types supportés

NSIMD supporte tous les types entiers signés et non signés existant sur 8 à 64 bits. Le support des nombres à virgule flottante se fait pour 16, 32, et 64 bits. Lorsqu'une architecture ne supporte pas nati-

1. https ://github.com/agenium-scale/nsimd

vement les nombres à virgule flottante sur 16 bits, leur fontionnement est émulé. NSIMD est également, à notre connaissance, la seule bibliothèque SIMD à proposer un support pour les nombres à virgule flottante sur 16 bits.

Interfaces de programmation proposées

NSIMD est compatible avec les standards C89, C++98, C++11 et C++14. Elle propose trois interfaces de programmation

ootnotehttps://agenium-scale.github.io/nsimd/index.html. La première est une interface C, et propose des opérateurs de la forme : `nsimd_{op}_ {ext}_{type}([args])`. Où `op` est le nom de l'opérateur utilisé, `ext` représente l'extension SIMD utilisée, et `type` représente le type utilisé, il peut être de la forme : `{i, u, f}{8, 16, 32, 64}`. Ce code est rendu générique par la définition de macros permettant d'écrire un code de la forme : `v{op}([args], {type})`. La définition de l'architecture cible à la compilation permet d'appeler les bonnes fonctions.

Les deux autres interfaces de programmation sont des interfaces C++. La première interface fournit des opérateurs génériques de la forme : `nsimd::{op}([args], type())`, le type à utiliser étant défini en paramètre. Enfin, la deuxième interface définit un type générique `nsimd::pack <{type}, {N}>` représentant un registre SIMD ainsi qu'un niveau de déroulage `N` (par défaut `1`), permettant un déroulage automatique des boucles. Les opérateurs sont sous la forme générique : `nsimd::op<{type}>([args])`.

Application au calcul d'éléments finis spectraux

EFISPEC3D

EFISPEC3D [2] est une bibliothèque logicielle de simulation sismique utilisant la méthode de éléments spectraux finis en 3D.

Cette bibliothèque est très répandue pour la simulation sismique, et l'optimisation de ses performances est un enjeu important pour accélérer la rapidité des simulations. Actuellement, les optimisations de cet algorithme reposent sur les capacités des processeurs à auto-vecotriser du code. Cependant, les gains en performances ne sont pas aussi bons qu'en vectorisant explicitement le code. Nous comparons ici les performances du cde de calcul d'EFISPEC3D vectorisées explicitement pour une architecture particulière, et une version équivalente vectorisée à l'aide de NSIMD.

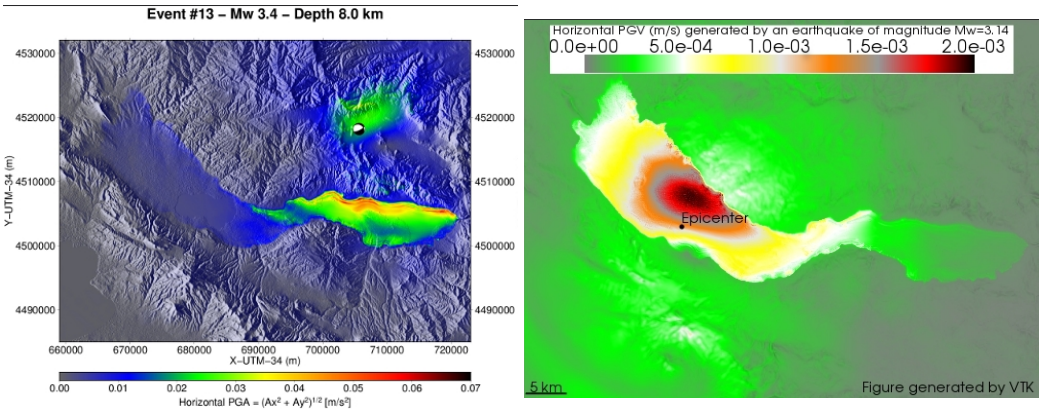


FIGURE 1 – *Algorithme EFISPEC3D.*

Compiler version

```
gcc (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
```

2. http://efispec.free.fr/

```
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Système d'exploitation

```
Linux caradigan 4.15.0-112-generic #113~16.04.1-Ubuntu SMP Fri Jul 10 04:38:45 UTC
    2020 aarch64 aarch64 aarch64 GNU/Linux
```

Informations sur l'architecture CPU

```
Architecture:          aarch64
Byte Order:            Little Endian
CPU(s):                64
On-line CPU(s) list:   0-63
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             16
NUMA node(s):          4
NUMA node0 CPU(s):     0-15
NUMA node1 CPU(s):     16-31
NUMA node2 CPU(s):     32-47
NUMA node3 CPU(s):     48-63
```

Informations sur la RAM

```
MemTotal:       131755592 kB
MemFree:        103095120 kB
MemAvailable:   128001480 kB
Buffers:          2095852 kB
Cached:          23486488 kB
SwapCached:            0 kB
Active:           8164932 kB
Inactive:        17506380 kB
Active(anon):      234732 kB
Inactive(anon):   1302264 kB
Active(file):     7930200 kB
Inactive(file):  16204116 kB
Unevictable:        2956 kB
Mlocked:            2956 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                60 kB
Writeback:             0 kB
AnonPages:         92028 kB
Mapped:           163456 kB
Shmem:           1446296 kB
Slab:            2238908 kB
SReclaimable:    1755288 kB
SUnreclaim:       483620 kB
KernelStack:       13216 kB
PageTables:         2292 kB
NFS_Unstable:          0 kB
```

```
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    65877796 kB
Committed_AS:    1825248 kB
VmallocTotal:   135290290112 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Information sur la bibliothèque standard

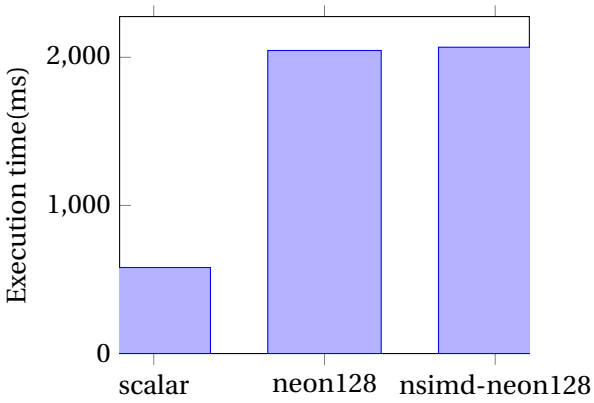
```
GNU C Library (Ubuntu GLIBC 2.23-0ubuntu11.2) stable release version 2.23, by
    Roland McGrath et al.
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
Compiled by GNU CC version 5.4.0 20160609.
Available extensions:
        crypt add-on version 2.1 by Michael Glad and others
        GNU Libidn by Simon Josefsson
        Native POSIX Threads Library by Ulrich Drepper et al
        BIND-8.2.3-T5B
libc ABIs: UNIQUE
For bug reporting instructions, please see:
<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>.
```

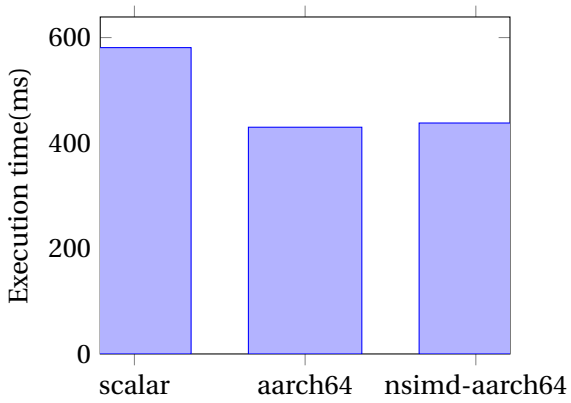
ewpage

Résultats

Benches results for neon128

Benches results for aarch64


Execution time(ms)
2,000
1,000
0
scalar
neon128
nsimd-neon128


Execution time(ms)
600
400
200
0
scalar
aarch64
nsimd-aarch64