GROMACS - Bug #1099

AVX_128_FMA acceleration broken with clang 3.1/3.2

12/29/2012 02:34 AM - Szilárd Páll

Status:	Closed		
Priority:	Normal		
Assignee:			
Category:	mdrun		
Target version:	4.6.3		
Affected version - extra info:	4.6-beta3	**Difficulty:**	uncategorized
Affected version:	N/A		

Description

clang 3.1 and 3.2 seems to generate incorrect code with GMX_CPU_ACCELERATION=AVX_128_FMA. Loads of regression tests fail both with group and verlet scheme.

Tested with vanilla (non-Apple) clang and 4.6-beta3 code on a Piledriver CPU.

Associated revisions

Revision 1dd009ab - 01/10/2013 09:24 PM - Erik Lindahl

Disallow the use of Clang to compile AVX_128_FMA

Clang apparently produces buggy code both for verlet kernels
and group kernels when the AVX_128_FMA acceleration is enabled.
Since it happens in two places it is unlikely to be a minor
bug, and since it covers versions up to the currently
released one, we don't have much choice but to disallow this
combination for now.
Fixes #1099.

Change-Id: If8346fa0a58f51a0ef6ac16e4801fe98ad1c73c9

Revision 34a402e7 - 05/30/2013 10:29 AM - Szilárd Páll

add workaround for clang on AMD with FMA

The clang bug which causes incorrect code to be generated on AMD
with FMA (Piledriver and Bulldozer) is in fact caused by an assembler
bug and can be worked around by switching to an external assembler.
This commit implements such the workaround for clang 3.1 and 3.2.
The bug was fix in clang version 3.2-svn-r173176:
http://llvm.org/bugs/show_bug.cgi?id=15040
Hence, no workaround is needed for clang 3.3 with AVX_128_FMA
acceleration.

This workaround is ineffective on Mac OS because there is no suitable
external assembler, but distinguishing between vanilla and Apple clang
requires more code than what's worth adding to cover the case of
AMD-based Hackintoshes (all Mac-s are Intel-based).

It is appropriate to amend only C and not C++ compilation for
release-4-6 branch, but when merging this to master branch, reconsider
the detail of the implementation.

Refs #1099

Change-Id: I835e0f2436ee33507514d3bc3980fcc227e2e36f

Revision c99ed93b - 06/27/2013 05:41 AM - Szilárd Páll

Adapted clang-AMD-FMA workaround for both C and C++

Based on 34a402e7bfb0c950edd7a7a624acf48334333a2d

The clang bug which causes incorrect code to be generated on AMD

with FMA (Piledriver and Bulldozer) is in fact caused by an assembler bug and can be worked around by switching to an external assembler. This commit implements such the workaround for clang 3.1 and 3.2. The bug was fix in clang version 3.2-svn-r173176: http://llvm.org/bugs/show_bug.cgi?id=15040
Hence, no workaround is needed for clang 3.3 with AVX_128_FMA acceleration.

This workaround is ineffective on Mac OS because there is no suitable external assembler, but distinguishing between vanilla and Apple clang requires more code than what's worth adding to cover the case of AMD-based Hackintoshes (all Mac-s are Intel-based).

Fixes #1099

Change-Id: Ice7c17a792fe257c4516628030b680d3b3b1a484

History

#1 - 01/10/2013 01:39 AM - Erik Lindahl

- Status changed from New to In Progress

#2 - 01/10/2013 01:39 AM - Erik Lindahl

- Assignee changed from Berk Hess to Erik Lindahl

#3 - 01/10/2013 03:41 AM - Roland Schulz

We should file a bug with the clang team before closing this bug.

#4 - 01/11/2013 12:38 PM - Erik Lindahl

- Status changed from In Progress to Feedback wanted

I think the obvious step is to have a real Gromacs release that we can point the clang team to, so I'm moving it to "feedback" since it's closed from Gromacs' p-o-v.

#5 - 01/17/2013 11:42 PM - Szilárd Páll

Erik Lindahl wrote:

> I think the obvious step is to have a real Gromacs release that we can point the clang team to, so I'm moving it to "feedback" since it's closed from Gromacs' p-o-v.

I agree. As soon as 4.6 is out, we should report it especially as clang, considering how young the compiler is, has an impressive performance on Intel CPUs getting very close in total performance to gcc/icc, in some parts of the code even slightly faster.

I volunteer for reporting the issue in the coming days, please let me know if you have any details I should mention in the report. I'll post a link to the report.

#6 - 01/18/2013 04:09 PM - Erik Lindahl

- Target version changed from 4.6 to 4.6.1

Changing target to 4.6.1, so we note it when we go through updated deadlines.

#7 - 02/15/2013 10:52 PM - Szilárd Páll

Filed: http://llvm.org/bugs/show_bug.cgi?id=15282

#8 - 04/29/2013 08:01 PM - Mark Abraham

- Assignee deleted (Erik Lindahl)

- Target version deleted (4.6.1)

- Affected version set to N/A

Possible solution for clang 3.2 noted on the above bug report

#9 - 04/29/2013 09:04 PM - Szilárd Páll

- Status changed from Feedback wanted to Closed

Today I've uploaded a change to gerrit which implements suggested workaround for clang 3.1/3.2 and white-lists clang 3.3.

Hence, I'll close this issue as the original bug is now solved properly.

#10 - 05/02/2013 04:34 PM - Szilárd Páll

Szilárd Páll wrote:

> Today I've uploaded a change to gerrit which implements suggested workaround for clang 3.1/3.2 and white-lists clang 3.3.

Note that the workaround is applicable to both 3.1 and 3.2 - this was confirmed by clang developers as well as testing, the current git code (4.6.2-dev) compiles and passes regressiontests.

#11 - 05/11/2013 09:04 AM - Mark Abraham

Confirmed also with Apple clang "4.1" based on LLVM 3.1. However the patch in gerrit fails to detect this case. #1039 had a similar issue, and the model for its solution seems to be http://redmine.gromacs.org/projects/gromacs/repository/revisions/bc34c5f9297f46c6520c48b21f190006d47abf3f

#12 - 05/11/2013 10:09 AM - Mark Abraham

Nope, that doesn't work either. Apple's clang "4.1" is evil and sets *clang_major* to 4

#13 - 05/11/2013 04:44 PM - Szilárd Páll

Unfortunately in the clang world it seems to be OK to claim any compiler version, hence checking version is not encouraged and instead features checks are recommended. Now, it might sound that this complicates things a lot. While this is true if we want to cover all clang derivatives which potentially claim to be some higher version than the upstream they branched off from, in practice ATM this only applies to Apple clang - at least I don't know of any other mainstream clang-based compiler. As there is no (official) AMD-based Mac, we could simply treat Mac OS as a platform where FMA4 is not supported in versions <=4.1. This check will probably not even need updating as the next clang version will probably include the FMA bugfix.

#14 - 05/12/2013 05:24 PM - Mark Abraham

- Status changed from Closed to In Progress

Szilárd Páll wrote:

> Unfortunately in the clang world it seems to be OK to claim any compiler version, hence checking version is not encouraged and instead features checks are recommended.

Indeed, but the "feature checks" don't check for correctness of implementation, so they are not very useful for the present case.

Now that we have a flexible and moderately portable SIMD API, some tests of it are reasonable to write (in master branch). This will catch this sort of issue when the user/developer/Jenkins runs "make check," because a test that triggers the buggy vfmaddps would probably fail.

> Now, it might sound that this complicates things a lot. While this is true if we want to cover all clang derivatives which potentially claim to be some higher version than the upstream they branched off from, in practice ATM this only applies to Apple clang - at least I don't know of any other mainstream clang-based compiler. As there is no (official) AMD-based Mac, we could simply treat Mac OS as a platform where FMA4 is not supported in versions <=4.1. This check will probably not even need updating as the next clang version will probably include the FMA bugfix.

Sounds reasonable. So avx_128_fma requires (!clang) or (clang and ((darwin and version > 4.1) or (!darwin and version > 3.3)). Further, in the darwin case we need to trigger the use of the external assembler, too.

#15 - 05/12/2013 05:25 PM - Mark Abraham

- Status changed from In Progress to Fix uploaded

#16 - 05/20/2013 10:21 PM - Szilárd Páll

Mark Abraham wrote:

> Sounds reasonable. So avx_128_fma requires (!clang) or (clang and ((darwin and version > 4.1) or (!darwin and version > 3.3)). Further, in the darwin case we need to trigger the use of the external assembler, too.

That won't work because (clang && darwin) does not imply Apple clang. I'll just simply always warn on Apple + AVX_128_FMA which will be very-very rare.

#17 - 05/21/2013 12:27 AM - Mark Abraham

Good point

#18 - 06/26/2013 01:49 AM - Mark Abraham

- Status changed from Fix uploaded to Resolved

- Target version set to 4.6.3

#19 - 06/27/2013 10:55 AM - Szilárd Páll

- % Done changed from 0 to 100

Applied in changeset c99ed93b144a0a880699537303521ae96f3e2199.

#20 - 09/18/2013 06:07 PM - Szilárd Páll

- Status changed from Resolved to Closed