nvcc host compiler sanity checks break with cmake >=2.8.10

The FindCUDA module included in CMake >=2.8.10 sets by default the CUDA_HOST_COMPILER variable without doing any sanity checks to make sure that the host compiler used is actually compatible with nvcc. With CUDA_HOST_COMPILER set our checks done in source/cmake/gmxManageNvccConfig.cmake get disabled. Therefore, with CMake >=2.8.10 the user does not get a warning about incompatible host compiler anymore.

Revision 770f143a - 05/05/2016 04:46 AM - Szilárd Páll

Remove CUDA host compiler consistency checks

Since CMake 2.8.10 the host compiler is set by CMake which effectively broke our consistency checks. However, these checks are hard to maintain, and even though CMake does not do any checks we are better off without this code.

This commit removed the checks, unconditionally sets the CUDA_HOST_COMPILER variable for CMake 2.8.9 and earlier - code that should be removed when CMAke 2.8.10 is required.

Fixes #1248

Change-Id: I6c08b59642dd3b5d18c5fe5ac454f19c75718f6a

History

#1 - 05/12/2013 02:52 AM - Szilárd Páll

The current checks do issue a warning along a lengthy explanation on why are we not auto-setting the nvcc host compiler (see gmxManageNvccConfig.cmake).

However, after some annoyingly lengthy debugging, I realized that FindCUDA included in CMake version 2.8.10 and later during the CUDA detection process automatically sets CUDA_HOST_COMPILER - which is essentially equivalent with the user passing this manually - which disables the checks. I don't have a good solution for this except checking whether the user passed CUDA_HOST_COMPILER before calling FindCUDA and handling the case of CMake >=2.8.10 separately. Ugly, but it should work.

#2 - 05/12/2013 05:42 PM - Mark Abraham

If there's behavioural changes in FindCUDA, then it sounds like we should get the version of FindCUDA we know works and include it in our source. That seems to me less ugly than littering our code with workarounds.

#3 - 05/14/2013 03:26 PM - Szilárd Páll

The problem is that as FindCUDA doesn't do any of the sanity checks on the compiler that we do and it will happily accept compilers that surely don't work with nvcc (e.g. clang) as well as MPI compiler wrappers - which can be problematic in some cases. If we want to keep the verbose user feedback, we need to decide whether we keep the FindCUDA >=2.8.10 behavior or discard the CUDA_HOST_COMPILER value and leave it unset in the cases when we do this ATM. Not sure what's best, but I don't think we should pull in CMake source code (again) just for the sake of this rather minor detail.

Note to self: we could potentially look for OMPI_CC and similar environment variables which can indicate what is the C compiler behind the MPI wrapper.

#4 - 05/14/2013 03:27 PM - Szilárd Páll
Btw, the actual issue has little to do with the bug description which is technically invalid. Should I create a new issue?

#5 - 05/14/2013 05:32 PM - Mark Abraham

Szilárd Páll wrote:

Btw, the actual issue has little to do with the bug description which is technically invalid. Should I create a new issue?

You should be able to edit my original post. Update - Change Properties - More

#6 - 05/17/2013 11:57 AM - Szilárd Páll

- Subject changed from clang and GPU does not work to nvcc host compiler sanity checks break with cmake >=2.8.10
- Category set to mdrun

#7 - 05/17/2013 12:05 PM - Szilárd Páll

- Description updated

#8 - 05/28/2013 11:37 PM - Szilárd Páll

- Status changed from New to In Progress

#9 - 06/26/2013 11:37 PM - Szilárd Páll

- Target version set to 4.6.x

Did 9765be25433d8275015f7a0198a2b312c8ed1 resolve this?

#10 - 06/26/2013 06:13 PM - Szilárd Páll

- Priority changed from Normal to Low

No, it did not, it made things work with the new cmake, but as a side-effect, with cmake >=v2.8.10 we don't check the host compiler (as with additional changes we can't distinguish between CUDA_HOST_COMPILER set by the user or by FindCUDA).

Not an big issue IMHO, so I'm dropping this to low priority.

#11 - 05/14/2014 10:39 AM - Rossen Apostolov

do you want to keep this open?

#12 - 06/13/2014 12:12 PM - Szilárd Páll

- Status changed from In Progress to Accepted

#13 - 06/19/2014 11:20 PM - Erik Lindahl

- Target version changed from 4.6.x to 5.x

#14 - 06/20/2015 10:25 AM - Gerrit Code Review Bot

Gerrit received a related patchset `1' for Issue #1248.
Uploader: Erik Lindahl (erik.lindahl@gmail.com)
Change-Id: I38a4a7f59b61ae6628d9d181ab52de27f65d35927
Gerrit URL: https://gerrit.gromacs.org/4749

#15 - 06/20/2015 10:27 AM - Erik Lindahl

- Status changed from Accepted to Fix uploaded

Not an exhaustive fix for the original CMake issues, but by testing the CUDA & host compiler combination I think we've done everything that is reasonable in Gromacs.

#16 - 04/03/2016 10:57 PM - Erik Lindahl

- Status changed from Fix uploaded to Accepted

This has now been open for three years, and nobody appears to have looked into it after we decided my quick-and-dirty fix didn't work (and had worse side effects).

At some point I guess we need to decide where we draw the line between the responsibility of GROMACS, Cmake, the operating system and the
user.
I fear we will end up with a huge mess of Cmake configurations that we then struggle to keep up to date if we try to write our own detection of the entire CUDA compiler configuration, so my preference would be that we drop this and simply tell the user it is their responsibility to pick a compatible cuda host compiler.

Looking at OS X, NVIDIA has deprecated gcc and only supports the default system clang compiler, so I suspect their future answer to this question is that they only support the default compiler on each system.

Thus, I would vote for dropping this issue and leaving it to the CMake module.

#17 - 04/04/2016 12:23 AM - Szilárd Páll
I agree, we need to make our life easier by offloading more burden on the user. I put a lot of work into checks, detection, warnings etc. to help users as much as possible in the initial GPU release, but indeed, this part of the code could use simplification.

However, CMake sets those variables without checking so the issue won't just go away without changing some code to remove the inconsistent way we handle things. I considered it too much effort/code complexity to implement the existing checks for the CMake >=2.8.10 and now I would prefer to not have to maintain these checks further - and these are getting outdated anyway.

The only solution I see is to remove the checks and just set the host compiler to whatever is detected. I'll keep the icc compatibility mode flag, although it leads to warnings with v15, but I don't have a better option right now (or am I missing something?).

#18 - 04/04/2016 12:35 AM - Szilárd Páll
- Target version changed from 5.x to 2016
- Affected version - extra info set to 5.0, 5.1

4.6-5.1 won't fix.

#19 - 04/04/2016 12:36 AM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1248.
Uploader: Szilárd Páll (pall.szilard@gmail.com)
Change-Id: I6c08b59642dd3b5d18c5fe5ac4541f9c75718f6a
Gerrit URL: https://gerrit.gromacs.org/5779

#20 - 04/04/2016 12:36 AM - Szilárd Páll
- Status changed from Accepted to Resolved

#21 - 05/05/2016 05:00 AM - Szilárd Páll
Applied in changeset 770f143a3960cdd169e4dfb975378e1c77e909d5.

#22 - 05/08/2016 12:50 PM - Erik Lindahl
- Status changed from Resolved to Closed