GROMACS - Bug #2582

Compilation issues with CUDA V9.1.85 and both gcc5 and gcc6

07/22/2018 10:10 AM - Semen Yesylevskyy

Status:	Blocked, need info		
Priority:	Normal		
Assignee:	Szilárd Páll		
Category:			
Target version:			
Affected version - extra info:		**Difficulty:**	uncategorized
Affected version:	2018.2		

Description

I'm trying to compile Gromacs 2018.2 on latest Linux Mint 19 with CUDA support.
Default gcc version is 7 which is not supported by CUDA, so I've installed gcc-5 and gcc-6 and tried what is written in the manual:

```
cmake ../gromacs -DCUDA_HOST_COMPILER=gcc-5 -DCMAKE_CXX_COMPILER=g++-5 -DCMAKE_C_COMPILER=gcc-5

-- Check for working NVCC/C compiler combination
-- Check for working NVCC/C compiler combination - broken
CMake Error at cmake/gmxManageGPU.cmake:291 (message):
  CUDA compiler does not seem to be functional.

cmake ../gromacs -DCUDA_HOST_COMPILER=gcc-6 -DCMAKE_CXX_COMPILER=g++-6 -DCMAKE_C_COMPILER=gcc-6
-- Check for working NVCC/C compiler combination
-- Check for working NVCC/C compiler combination - broken
CMake Error at cmake/gmxManageGPU.cmake:291 (message):
  CUDA compiler does not seem to be functional.
```

Here is the version of nvcc:

```
$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri_Nov__3_21:07:56_CDT_2017
Cuda compilation tools, release 9.1, V9.1.85
```

I only managed to compile after manually linking gcc version 5 to cuda:

```
sudo ln -s /usr/bin/gcc-5 /usr/lib/cuda/bin/gcc

cmake ../gromacs -DCUDA_HOST_COMPILER=g++-5 -DCMAKE_CXX_COMPILER=g++-5 -DCMAKE_C_COMPILER=gcc-5
...
-- Configuring done
-- Generating done
-- Build files have been written to: /home/semen/install/gromacs-git/gromacs-build
```

With gcc-6 it still doesn't work.

I think the procedure of compiling with CUDA should be written in much more details in the docs. It seems that the issues with compiler mismatch are not as easy as setting CUDA_HOST_COMPILER.

History

#1 - 07/23/2018 05:16 PM - Szilárd Páll

- Status changed from New to Blocked, need info

You've not included a full output, so I can only guess what's going wrong.
Firstly, you do not need to pass CUDA_HOST_COMPILER unless you want to use one that's not the same as the default (the C++ compiler used in the build). Secondly, if you do pass it, you need to pass the *absolute* path to the C++ compiler binary, otherwise it will be assumed that you are passing a path relative to the build tree.

Consequently, unless I'm mistaken (and assuming that your gnu toolchains do work), either of your above command lines will just work if you omit the erroneously passed -DCUDA_HOST_COMPILER.

#2 - 07/23/2018 05:16 PM - Szilárd Páll

- Assignee set to Szilárd Páll

#3 - 07/23/2018 06:42 PM - Semen Yesylevskyy

Yes, you are right - the problem vanishes when CUDA_HOST_COMPILER is not passed.

With gcc-5 it compiles just fine. However, there is still a problem with gcc-6:

```
$ cmake ../gromacs -DCMAKE_CXX_COMPILER=g++-6 -DCMAKE_C_COMPILER=gcc-6

// Works just fine, no errors.

$ make
[  0%] Building NVCC (Device) object src/gromacs/gpu_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/gpu_utilste
st_cuda_generated_devicetransfers.cu.o
Scanning dependencies of target mdrun_objlib
[  0%] Building CXX object src/programs/mdrun/tests/CMakeFiles/mdrun_test_objlib.dir/energyreader.cpp.o
[  0%] Building C object src/gromacs/CMakeFiles/tng_io_obj.dir/__/external/tng_io/src/compression/xtc2.c.o
[  0%] Building C object src/gromacs/CMakeFiles/tng_io_obj.dir/__/external/tng_io/src/compression/tng_compress
.c.o
[  0%] Generating git version information
[  0%] Built target view_objlib
[  1%] Built target tng_io_zlib
[  1%] Building C object src/gromacs/CMakeFiles/tng_io_obj.dir/__/external/tng_io/src/lib/tng_io.c.o
[  1%] Building C object src/gromacs/CMakeFiles/tng_io_obj.dir/__/external/tng_io/src/compression/xtc3.c.o
In file included from /usr/include/crt/math_functions.h:8835:0,
                 from /usr/include/crt/common_functions.h:271,
                 from /usr/include/common_functions.h:50,
                 from /usr/include/cuda_runtime.h:115,
                 from <command-line>:0:
/usr/include/c++/6/cmath:45:23: fatal error: math.h: No such file or directory
 #include_next <math.h>
                       ^
compilation terminated.
CMake Error at gpu_utilstest_cuda_generated_devicetransfers.cu.o.cmake:207 (message):
  Error generating
  /home/semen/install/gromacs-git/gromacs-build/src/gromacs/gpu_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/
/./gpu_utilstest_cuda_generated_devicetransfers.cu.o

src/gromacs/gpu_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/build.make:63: recipe for target 'src/gromacs/gp
u_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/gpu_utilstest_cuda_generated_devicetransfers.cu.o' failed
make[2]: *** [src/gromacs/gpu_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/gpu_utilstest_cuda_generated_devic
etransfers.cu.o] Error 1
CMakeFiles/Makefile2:3491: recipe for target 'src/gromacs/gpu_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/al
l' failed
make[1]: *** [src/gromacs/gpu_utils/tests/CMakeFiles/gpu_utilstest_cuda.dir/all] Error 2
make[1]: *** Waiting for unfinished jobs....
```

I assume that is CMake is happy with gcc-6 it is supposed to compile but it doesn't. So there is still something wrong with detecting compatible compiler for CUDA, apparently gcc-6.4 is assumed to be Ok, but it is not.

```
$ gcc-6 --version
gcc-6 (Ubuntu 6.4.0-17ubuntu1) 6.4.0 20180424
Copyright (C) 2017 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.
```

#4 - 07/23/2018 07:45 PM - Szilárd Páll

There is something strange about your gcc 6 install, does it work otherwise, e.g. with GMX_GPU=OFF?

Is there a strong reason why you don't use the latest CUDA, version 9.2 -- it supports gcc 7 and it might well work better?

To clarify, the build system only executes a couple of sanity checks to make sure that the most common errors are detected early, at cmake-time rather than during compile-time; e.g. that the chosen C++ compiler is compatible with CUDA. It however does not and can not attempt to provide a bulletproof test of whether the combination of an arbitrary C++ compiler and CUDA installation will succeed -- generally that's not possible without doing the actual full compilation. Some issues with compiler toolchains are just very hard to detect especially with combinations of uncommon (and officially unsupported) combinations of software. For that reason, while the false negative sanity check could be considered a bug, unless it proves to be a common failure that we missed to test for, it will not be considered as such.

As a side-note: FWIW only the following combinations are "officially" supported by NVIDIA for CUDA 9.1: https://docs.nvidia.com/cuda/archive/9.1/cuda-installation-guide-linux/index.html#system-requirements: neither Linux Mint 19 nor gcc 6.4 are on that list. That's not to say that it it will not generally work, but there unpleasant surprises can crop up.

#5 - 07/23/2018 07:46 PM - Szilárd Páll

As a side-note, the gcc 6 issue is likely caused by CMake / misuse of -isystem, ref: https://stackoverflow.com/questions/37218953/isystem-on-a-system-include-directory-causes-errors