GROMACS - Bug #2958

Compiling master (to become 2020) using CUDA 9.0

05/23/2019 04:26 PM - Magnus Lundborg

Status:	New		
Priority:	Normal		
Assignee:			
Category:	build system		
Target version:	2020.2		
Affected version - extra info:	master (current commit f457e9a4ed4341a478e8d4c271d41f99e6f7ae24)	**Difficulty:**	uncategorized
Affected version:	git master		

Description

When compiling master with CUDA 9.0 (gcc-5 or gcc-6) I get the following type of errors:

/usr/lib/gcc/x86_64-linux-gnu/6/include/avx512fintrin.h(9220): error: argument of type "const void *" is incompatible with parameter of type "const float *"
...
92 errors detected in the compilation of "/tmp/tmpxft_00002518_00000000-13_nbnxm_cuda_data_mgmt.compute_70.cpp1.ii".
CMake Error at libgromacs_generated_nbnxm_cuda_data_mgmt.cu.o.cmake:266 (message):
Error generating file
/home/magnus/install/gromacs_master_test/src/gromacs/CMakeFiles/libgromacs.dir/nbnxm/cuda/./libgromacs_generated_nbnxm_cuda_data_mgmt.cu.o

...

when compiling with gcc-6 I also get a bunch of:

src/gromacs/CMakeFiles/libgromacs.dir/build.make:63: recipe for target 'src/gromacs/CMakeFiles/libgromacs.dir/nbnxm/cuda/libgromacs_generated_nbnxm_cuda.cu.o' failed
make[2]: *** [src/gromacs/CMakeFiles/libgromacs.dir/nbnxm/cuda/libgromacs_generated_nbnxm_cuda.cu.o] Error 1
/usr/include/c++/6/tuple: In instantiation of ‘static constexpr bool std::_TC<<anonymous>, *Elements>::_MoveConstructibleTuple() [with _UElements = {std::tuple<int&, int&, int&>}; bool <anonymous> = true; _Elements = {int&, int&, int&}]’:*
/usr/include/c++/6/tuple:626:248: required by substitution of ‘template<class ... _UElements, typename std::enable_if<(((std::_TC<(sizeof... (_UElements) 1), int&, int&, int&>::_NotSameTuple<_UElements ...>() && std::_TC<(1ul sizeof... (_UElements)), int&, int&, int&>::_MoveConstructibleTuple<_UElements ...>()) && std::_TC<(1ul sizeof... (_UElements)), int&, int&, int&>::_ImplicitlyMoveConvertibleTuple<_UElements ...>()) && (3ul >= 1)), bool>::type <anonymous> > constexpr std::tuple< <template-parameter-1-1> >::tuple(_UElements&& ...) [with _UElements = {std::tuple<int&, int&, int&>};
typename std::enable_if<(((std::_TC<(sizeof... (_UElements) 1), int&, int&, int&>::_NotSameTuple<_UElements ...>() && std::_TC<(1ul sizeof... (_UElements)), int&, int&, int&>::_MoveConstructibleTuple<_UElements ...>()) && std::_TC<(1ul sizeof... (_UElements)), int&, int&, int&>::_ImplicitlyMoveConvertibleTuple<_UElements ...>()) && (3ul >= 1)), bool>::type <anonymous> = <missing>]’
/usr/include/c++/6/tuple:1545:43: required from ‘constexpr std::tuple<_Elements& ...> std::tie(_Elements& ...) [with _Elements = {int, int, int}]’
/home/magnus/development/gromacs/src/gromacs/mdlib/lincs_cuda_impl.cu:601:24: required from here
/usr/include/c++/6/tuple:483:67: error: mismatched argument pack lengths while expanding ‘std::is_constructible<_Elements, _UElements&&>’
return __and<is_constructible<_Elements, _UElements&&>...>::value;

History

#1 - 05/23/2019 04:36 PM - Magnus Lundborg

With CUDA-9.2 I can compile with gcc-7, but not gcc-6 (I haven't tested gcc-5).

#2 - 09/06/2019 02:10 PM - Paul Bauer

@Magnus, is this still an issue?

#3 - 09/06/2019 02:26 PM - Magnus Lundborg

Yes, I get the same error with the current master using CUDA-9.0 and gcc-6 at least.

#4 - 09/24/2019 03:25 PM - Paul Bauer

- Target version changed from 2020-beta1 to 2020-beta2

#5 - 10/14/2019 09:15 PM - Szilárd Páll

no progress, suggest bump

#6 - 11/01/2019 03:24 PM - Paul Bauer

- Target version changed from 2020-beta2 to 2020-beta3

bump

#7 - 12/02/2019 01:12 PM - Paul Bauer

- Target version changed from 2020-beta3 to 2020-rc1

bump

#8 - 12/20/2019 08:24 AM - Paul Bauer

- Target version changed from 2020-rc1 to 2020

again, this needs fixing for 2020!

#9 - 12/20/2019 09:51 AM - Magnus Lundborg

I agree. CUDA 9.2 is required to use gcc-7. This is worrying since CUDA 9.0 (or possibly 9.1) is the version available in, e.g. Ubuntu 18.04.

#10 - 12/27/2019 04:27 PM - Paul Bauer

- Target version changed from 2020 to 2020.1

#11 - 02/25/2020 03:37 PM - Szilárd Páll

If we have a known to be broken combination of CUDA and gcc (and we have no known workaround) I suggest we add a cmake-time check and suggest that users download a more recent CUDA release (that can be used with gcc 7 or 8).

#12 - 03/02/2020 03:57 PM - Paul Bauer

- Target version changed from 2020.1 to 2020.2