GROMACS - Bug #2273

CUDA CC 2.0 issue

10/13/2017 06:44 PM - Mark Abraham

Status:	Closed		
Priority:	Normal		
Assignee:			
Category:	mdrun		
Target version:	2018		
Affected version - extra info:	master HEAD	**Difficulty:**	uncategorized
Affected version:	2016.4		

Description

Using either master HEAD, release-2016 HEAD, or tag v2016:

If I target compute and sm 20 (with -DGMX_CUDA_TARGET_SM=20 -DGMX_CUDA_TARGET_COMPUTE=20) then by default we get a single CUDA compilation unit (since that's the only thing that can work). The regressiontests pass, but we we have an issue, e.g.

```
$ bin/mdrun-test --gtest_filter=\*Swap\*
...
Running on 1 node with total 4 cores, 8 logical cores, 2 compatible GPUs
Hardware detected:
  CPU info:
    Vendor: Intel
    Brand:  Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
    SIMD instructions most likely to fit this hardware: AVX_256
    SIMD instructions selected at GROMACS compile time: AVX_256

  Hardware topology: Full, with devices
  GPU info:
    Number of GPUs detected: 2
    #0: NVIDIA GeForce GTX 960, compute cap.: 5.2, ECC:  no, stat: compatible
    #1: NVIDIA GeForce GTX 660 Ti, compute cap.: 3.0, ECC:  no, stat: compatible

Reading file /home/marklocal/git/r2016/build-cmake-gcc-gpu-cc20-debug/src/programs/mdrun/tests/Tes
ting/Temporary/CompelTest_SwapCanRun.tpr, VERSION 2016.5-dev-20170923-d36730ca3 (single precision)
Using 1 MPI thread
Using 1 OpenMP thread

1 GPU user-selected for this run.
Mapping of GPU ID to the 1 PP rank in this node: 0

NOTE: Thread affinity setting failed. This can cause performance degradation.
      If you think your settings are correct, ask on the gmx-users list.

SWAP: Determining initial numbers of ions per compartment.
SWAP: Setting pointers for checkpoint writing
SWAP: Channel 0 flux history for ion type NA+ (charge 1): 0 molecules
SWAP: Channel 1 flux history for ion type NA+ (charge 1): 0 molecules
SWAP: Channel 0 flux history for ion type CL- (charge -1): 0 molecules
SWAP: Channel 1 flux history for ion type CL- (charge -1): 0 molecules
starting mdrun 'Channel_coco in octane membrane'
2 steps,      0.0 ps.

-------------------------------------------------------
Program:     mdrun-test, version 2016.5-dev-20170923-d36730ca3
Source file: src/gromacs/mdlib/nbnxn_cuda/nbnxn_cuda.cu (line 633)

Fatal error:
cudaStreamSynchronize failed in cu_blockwait_nb: an illegal memory access was
```

```
encountered

For more information and tips for troubleshooting, please check the GROMACS
website at http://www.gromacs.org/Documentation/Errors
-------------------------------------------------------
```

If I target compute and sm 30 then, by default, I get multiple CUDA compilation units and there is no issue.
If I target compute and sm 30 and set -DGMX_CUDA_NB_SINGLE_COMPILATION_UNIT=on, then there is no issue.

So it looks to me like something in the CC 2.0 support is broken, or at least not properly used by the mdrun-test code. I'll try to bisect a bit more and see what I learn.

The absence of a reported bug does suggest that there is not much use of release-2016 on CC 2.0, and we should consider removing support for CC 2.0 for GROMACS 2017. This would simplify our texture and CMake code, and remove the question of whether someone should try to cover this case in Jenkins. Clearly nobody has prioritized doing or automating testing on this old setup. Note that NVIDIA has already deprecated those compilation targets in nvcc (and we suppress the warning). If we go this path, then I suggest we don't bother trying to fix release-2016, and if someone later has an issue, suggest they use an even earlier version.

Associated revisions

Revision 29ba77b8 - 10/31/2017 08:19 PM - Szilárd Páll

Check CUDA available/compiled code compatibility

Added an early check to detect when the gmx binary does not embed code
compatible with the GPU device it tries to use nor does it have PTX that
could have been JIT-ed.

Additionally, if the user manually sets GMX_CUDA_TARGET_COMPUTE=20 and
no later SM or COMPUTE but runs on >2.0 hardware, we'd be executing
JIT-ed Fermi kernels with incorrect host-side code assumptions
(e.g amount of shared memory allocated or texture type).
This change also prevents such cases.

Fixes #2273

Change-Id: I5472b1a33e584a75f451e21e9fd25992633fbea9

History

#1 - 10/16/2017 05:04 PM - Szilárd Páll

The problem in practice is that you're requesting only sm_20 and compute_2.0 (i.e. SASS for the 2.0 arch and PTX for the 2.0 virtual arch) to be embedded in the binary but this binary is then run on a CC 5.2 device. JIT compilation is in this case is considered valid from 2.0 virt. arch to sm_52. However, in practice this is not valid as the two kernel flavors have different resource needs (see calc_shmem_required_nonbonded()). However, as JIT silently accepts forward-compatible compilation (of already pre-processed code, otherwise the #error in nbnxn_cuda_kernel_fermi.cuh would get trigegred) and therefore at runtime we have no way to detect that the wrong kernel is getting dispatched to a newer GPU.

This is a very much corner-case that I do not think it's worth catering for, the only case where such an error can be reasonably triggered is if a user compiled a binary many years ago for some reason only for Fermi and then they upgrade their GPU.

I'm not even sure a workaround is possible as we can't query at runtime which kernels were compiled in without string parsing command line options (which seems like an overkill).

#2 - 10/16/2017 05:44 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2273.
Uploader: Szilárd Páll (pall.szilard@gmail.com)
Change-Id: gromacs~master~I5472b1a33e584a75f451e21e9fd25992633fbea9
Gerrit URL: https://gerrit.gromacs.org/7043

#3 - 10/20/2017 11:15 AM - Aleksei Iupinov

Just ran same 2.0/3.0 procedures on ewald-test with CUDA 6.5.
In compute_20 build, the PME spread tests fail only - also with exact same error (an illegal memory access was encountered during stream synchronization).
The tests don't fail if textures are disabled.
Spread is the only PME kernel which uses textures, and it's the only arch-dependent feature of the kernel.
Clearly, the problem lies with texrefs. Maybe we misuse them, or they are not supposed to be used on (CC >= 3.0) hardware at all.
So the general solution is needed.

#4 - 10/22/2017 04:38 PM - Szilárd Páll

The source of the issue is still the same, I think: the host code only takes into account at runtime what is the CC of the device in use and does not

consider that the kernel may be compiled from PTX targeting a virt arch that does not support a certain feature (be it texobj or shuffle reduction).

#5 - 10/23/2017 10:56 AM - Mark Abraham

So, the host code needs to have more flexible behaviour, so that the PTX will always work. Further, it sounds like we need that in order that the compilation for multiple specific archs is always compatible with the host behaviour.

For 2016, either we need a build mode that supports CC 2.0 that always works (and doesn't break other support), or we need to remove 2.0 support. Similarly for 2017, but I would further suggest we don't implement 2.0 support for new functionality.

#6 - 10/31/2017 09:23 PM - Szilárd Páll

- Status changed from New to Resolved

Applied in changeset 29ba77b8483f803766806b4f6987aeeef00747e5.

#7 - 10/31/2017 09:45 PM - Mark Abraham

- Target version set to 2018

We haven't (and probably won't?) attempted to fix this for release-2016

#8 - 11/28/2017 05:59 PM - Mark Abraham

- Status changed from Resolved to Closed