GROMACS - Bug #2762

incorrect results with Ubuntu 18.04 / glibc 2.27 (?) and >20 threads

11/15/2018 11:04 PM - Dmytro Kovalskyy

Status:	Blocked, need info		
Priority:	Normal		
Assignee:			
Category:	mdrun		
Target version:			
Affected version - extra info:	2019.x	**Difficulty:**	hard
Affected version:	2018.3		

Description

Hi,

Setup:
Dual Xeon NVODOA P5000
Ubuntu 18.04 4.15.0-36-generic Gromacs 2018-3, CUDA 10
Gromacs is compiled with
cmake .. -DGMX_GPU=ON -DGMX_USE_NVML=ON

The problem:
MD with GPU crashes with following error:

```
Program:     gmx mdrun, version 2018.3
Source file: src/gromacs/gpu_utils/cudautils.cu (line 110)

Fatal error:
HtoD cudaMemcpyAsync failed: invalid argument
```

When gmx mdrun -deffnm md200ns -v -nb cpu
or
gmx mdrun -deffnm md200ns -v -nb gpu -pme cpu

then MD goes as expected.

When Gromacs is compiled with Debug option, then MD run goes as expected with no additional options
i.e.
gmx mdrun -deffnm md200ns -v

```
1 GPU auto-selected for this run.
Mapping of GPU IDs to the 2 GPU tasks in the 1 rank on this node:
PP:0,PME:0
```

TPR and log files are attached.

thank you

Associated revisions

Revision 9f45a4be - 05/27/2019 11:04 AM - Berk Hess

Work around gcc 7 avx512 bug

Due to an avx512 loop vectorization bug in gcc 7, many non-bonded
interactions could be ignored when running with more than 16 OpenMP
threads.

Fixes #2762

Change-Id: I3e03fde7114542bbd43069166a6c74937fc0f986

History

#1 - 11/16/2018 01:29 AM - Mark Abraham

- File quick.log added

- File quick.out added

- File quick.tpr added

Thanks for the report and files.

So far I can't reproduce the issue, even on a rather similar build and machine. I build 2018.3 from the git repo, with NVML support and as close as I could to your setup, but none of the variations that I tried did anything out of the ordinary.

Is there another card of the same type as you are using? Or even a different type? What happens with mdrun -notunepme? What happens with a build that uses cmake -DCMAKE_BUILD_TYPE=RelWithDebug? What happens with the the 2019-beta2 build (some clean up we did in the PME on GPU code might mean there is no error, or we get a better clue where the error happens)?

#2 - 11/16/2018 02:02 AM - Szilárd Páll

Can you post the output of nvidia-smi -q? Perhaps the device is not in default compute mode.

#3 - 11/16/2018 09:32 PM - Dmytro Kovalskyy

Szilárd Páll wrote:

> Can you post the output of nvidia-smi -q? Perhaps the device is not in default compute mode.

```
nvidia-smi -q

==============NVSMI LOG==============

Timestamp                       : Fri Nov 16 14:20:47 2018
Driver Version                  : 410.73
CUDA Version                    : 10.0

Attached GPUs                   : 1
GPU 00000000:D5:00.0
Product Name                    : Quadro P5000
Product Brand                   : Quadro
Display Mode                    : Enabled
Display Active                  : Enabled
Persistence Mode                : Enabled
Accounting Mode                 : Disabled
Accounting Mode Buffer Size     : 4000
Driver Model
Current                         : N/A
Pending                         : N/A
Serial Number                   : 0320118044152
GPU UUID                        : GPU-11d59047-c90b-2322-3eba-e259dcf31a80
Minor Number                    : 0
VBIOS Version                   : 86.04.69.00.3F
MultiGPU Board                  : No
Board ID                        : 0xd500
GPU Part Number                 : 900-5G413-0100-000
Inforom Version
Image Version                   : G413.0500.00.02
OEM Object                      : 1.1
ECC Object                      : 4.1
Power Management Object         : N/A
GPU Operation Mode
Current                         : N/A
Pending                         : N/A
GPU Virtualization Mode
Virtualization mode             : None
IBMNPU
Relaxed Ordering Mode           : N/A
PCI
Bus                             : 0xD5
Device                          : 0x00
Domain                          : 0x0000
Device Id                       : 0x1BB010DE
Bus Id                          : 00000000:D5:00.0
Sub System Id                   : 0x11B21028
```

```
GPU Link Info
PCIe Generation
Max                 : 3
Current             : 1
Link Width
Max                 : 16x
Current             : 16x
Bridge Chip
Type                    : N/A
Firmware                : N/A
Replays since reset         : 0
Tx Throughput               : 0 KB/s
Rx Throughput               : 0 KB/s
Fan Speed                     : 26 %
Performance State               : P8
Clocks Throttle Reasons
Idle                        : Active
Applications Clocks Setting : Not Active
SW Power Cap                : Not Active
HW Slowdown                 : Not Active
HW Thermal Slowdown     : Not Active
HW Power Brake Slowdown : Not Active
Sync Boost                  : Not Active
SW Thermal Slowdown         : Not Active
Display Clock Setting       : Not Active
FB Memory Usage
Total                       : 16256 MiB
Used                        : 1291 MiB
Free                        : 14965 MiB
BAR1 Memory Usage
Total                       : 256 MiB
Used                        : 30 MiB
Free                        : 226 MiB
Compute Mode                    : Default
Utilization
Gpu                         : 0 %
Memory                        : 6 %
Encoder                       : 0 %
Decoder                       : 0 %
Encoder Stats
Active Sessions             : 0
Average FPS                 : 0
Average Latency             : 0
FBC Stats
Active Sessions             : 0
Average FPS                 : 0
Average Latency             : 0
Ecc Mode
Current                     : Disabled
Pending                       : Disabled
ECC Errors
Volatile
Single Bit
Device Memory       : N/A
Register File       : N/A
L1 Cache            : N/A
L2 Cache            : N/A
Texture Memory      : N/A
Texture Shared      : N/A
CBU                 : N/A
Total               : N/A
Double Bit
Device Memory       : N/A
Register File       : N/A
L1 Cache            : N/A
L2 Cache            : N/A
Texture Memory      : N/A
Texture Shared      : N/A
CBU                 : N/A
Total               : N/A
Aggregate
Single Bit
Device Memory       : N/A
Register File       : N/A
L1 Cache            : N/A
```

```
L2 Cache            : N/A
Texture Memory      : N/A
Texture Shared      : N/A
CBU                 : N/A
Total               : N/A
Double Bit
Device Memory       : N/A
Register File       : N/A
L1 Cache            : N/A
L2 Cache            : N/A
Texture Memory      : N/A
Texture Shared      : N/A
CBU                 : N/A
Total               : N/A
Retired Pages
Single Bit ECC              : N/A
Double Bit ECC              : N/A
Pending                     : N/A
Temperature
GPU Current Temp            : 33 C
GPU Shutdown Temp           : 99 C
GPU Slowdown Temp           : 96 C
GPU Max Operating Temp      : N/A
Memory Current Temp         : N/A
Memory Max Operating Temp   : N/A
Power Readings
Power Management            : Supported
Power Draw                  : 12.60 W
Power Limit                 : 180.00 W
Default Power Limit         : 180.00 W
Enforced Power Limit        : 180.00 W
Min Power Limit             : 90.00 W
Max Power Limit             : 180.00 W
Clocks
Graphics                    : 139 MHz
SM                          : 139 MHz
Memory                      : 405 MHz
Video                       : 544 MHz
Applications Clocks
Graphics                    : 1733 MHz
Memory                      : 4513 MHz
Default Applications Clocks
Graphics                    : 1607 MHz
Memory                      : 4513 MHz
Max Clocks
Graphics                    : 1733 MHz
SM                          : 1733 MHz
Memory                      : 4513 MHz
Video                       : 1569 MHz
Max Customer Boost Clocks
Graphics                    : 1733 MHz
Clock Policy
Auto Boost                  : N/A
Auto Boost Default          : N/A
Processes
......
```

#4 - 11/16/2018 10:07 PM - Dmytro Kovalskyy

Mark Abraham wrote:

> Thanks for the report and files.
>
> So far I can't reproduce the issue, even on a rather similar build and machine. I build 2018.3 from the git repo, with NVML support and as close as I could to your setup, but none of the variations that I tried did anything out of the ordinary.
>
> Is there another card of the same type as you are using?

I have only one card for now and using it for graphics now. But I have ordered a dedicated card, so once I have it I will split graphics from GPU. I will report once I have it installed.

> Or even a different type? What happens with mdrun -notunepme?

I got following error
gmx mdrun -deffnm md200ns -v -notunepme

```
Program:     gmx mdrun, version 2018.3
Source file: src/gromacs/gpu_utils/cudautils.cu (line 110)

Fatal error:
HtoD cudaMemcpyAsync failed: invalid argument
```

> What happens with a build that uses cmake -DCMAKE_BUILD_TYPE=RelWithDebug?

In this case MD runs as expected no additional options are supplied. But I see only 36 cores are busy (i.e. virtual cores are not loaded). Not sure how GPU is loaded.

```
Using 1 MPI thread
Using 36 OpenMP threads 

1 GPU auto-selected for this run.
Mapping of GPU IDs to the 2 GPU tasks in the 1 rank on this node:
  PP:0,PME:0
```

> What happens with the the 2019-beta2 build (some clean up we did in the PME on GPU code might mean there is no error, or we get a better clue where the error happens)?

gmx mdrun -deffnm md200ns -v

```
Using 1 MPI thread
Using 36 OpenMP threads 

1 GPU auto-selected for this run.
Mapping of GPU IDs to the 2 GPU tasks in the 1 rank on this node:
  PP:0,PME:0
PP tasks will do (non-perturbed) short-ranged interactions on the GPU
PME tasks will do all aspects on the GPU

Back Off! I just backed up md200ns.xtc to ./#md200ns.xtc.20#

Back Off! I just backed up md200ns.edr to ./#md200ns.edr.20#

WARNING: This run will generate roughly 7672 Mb of data

starting mdrun '2 SYMMETRICAL 'FOOTBALL' COMPLEX in water'
100000000 steps, 200000.0 ps.

-------------------------------------------------------
Program:     gmx mdrun, version 2019-beta2
Source file: src/gromacs/gpu_utils/devicebuffer.cuh (line 131)
Function:    copyToDeviceBuffer(ValueType**, const ValueType*, size_t, size_t, CommandStream, GpuApiCallBehavi
or, CommandEvent*)::<lambda()> [with ValueType = nbnxn_cj4_t]

Assertion failed:
Condition: stat == cudaSuccess
Asynchronous H2D copy failed
```

but with
gmx mdrun -deffnm md200ns -v -nb gpu -pme cpu

```
Back Off! I just backed up md200ns.log to ./#md200ns.log.22#
Reading file md200ns.tpr, VERSION 2018.3 (single precision)
Note: file tpx version 112, software tpx version 116
Changing nstlist from 20 to 80, rlist from 0.931 to 1.049

Using 12 MPI threads
Using 6 OpenMP threads per tMPI thread
On host didesk 1 GPU auto-selected for this run.
Mapping of GPU IDs to the 12 GPU tasks in the 12 ranks on this node:
  PP:0,PP:0,PP:0,PP:0,PP:0,PP:0,PP:0,PP:0,PP:0,PP:0,PP:0,PP:0
PP tasks will do (non-perturbed) short-ranged and most bonded interactions on the GPU

Back Off! I just backed up md200ns.xtc to ./#md200ns.xtc.22#

Back Off! I just backed up md200ns.edr to ./#md200ns.edr.22#
```

```
WARNING: This run will generate roughly 7672 Mb of data

NOTE: DLB will not turn on during the first phase of PME tuning
starting mdrun '2 SYMMETRICAL 'FOOTBALL' COMPLEX in water'
100000000 steps, 200000.0 ps.
step  160: timed with pme grid 52 60 72, coulomb cutoff 0.900: 608.9 M-cycles
step  320: timed with pme grid 48 56 64, coulomb cutoff 0.944: 583.1 M-cycles
step  480: timed with pme grid 44 52 60, coulomb cutoff 1.023: 569.3 M-cycles
step  640: timed with pme grid 40 44 56, coulomb cutoff 1.155: 587.9 M-cycles
^C

Received the INT signal, stopping within 80 steps

 Dynamic load balancing report:
 DLB was locked at the end of the run due to unfinished PP-PME balancing.
 Average load imbalance: 3.2%.
 The balanceable part of the MD step is 52%, load imbalance is computed from this.
 Part of the total run time spent waiting due to load imbalance: 1.6%.

               Core t (s)   Wall t (s)        (%)
       Time:      589.208        8.184     7199.4
                 (ns/day)    (hour/ns)
Performance:       15.223        1.577
```

#5 - 11/19/2018 01:05 PM - Szilárd Páll

(Made the parent a bit more readable.)

The nvidia-smi output doesn't show anything of interest.

#6 - 11/19/2018 01:05 PM - Szilárd Páll

- Subject changed from GPU iisue with CUDA 10 to GPU isue with CUDA 10

#7 - 11/20/2018 04:26 AM - Mark Abraham

That the type of failing transfer is nbnxn_cj4_t in 2019-beta2 may be a useful clue. (And perhaps confirmation that giving a fatal error at the point something fails is an error handling mechanism that we should prioritize replacing.) Is one of the the first transfers (or first of a stage) a struct of that type?

#8 - 11/20/2018 02:17 PM - Szilárd Páll

Mark Abraham wrote:

> That the type of failing transfer is nbnxn_cj4_t in 2019-beta2 may be a useful clue.

It may, however the rewrite of the transfer function ~~butchered~~ introduced a regression in the error reporting by removing the error code and string, so we have no clue which error occurred.

> (And perhaps confirmation that giving a fatal error at the point something fails is an error handling mechanism that we should prioritize replacing.)

Not sure how would that help. We check errors, so they *should not* propagate; if we'd let errors propagate, it would be slightly harder to trace them.

> Is one of the the first transfers (or first of a stage) a struct of that type?

It's the second at search step. Nothing special about it.
source:src/gromacs/mdlib/nbnxn_cuda/nbnxn_cuda_data_mgmt.cu#L554

#9 - 11/20/2018 02:20 PM - Szilárd Páll

Dmytro, long shot, but have you tried to run a stress-test on the card, e.g. cuda_memtest is useful for such purposes.

#10 - 11/22/2018 01:56 AM - Dmytro Kovalskyy

I can not compile cuda_memtest. First I got error

$ /usr/local/tmp/cuda_memtest-dev $ make
nvcc -c -arch sm_10 -DSM_10 -O3 -I. -I/usr/local/cuda/include -o cuda_memtest.o cuda_memtest.cu
nvcc fatal   : Value 'sm_10' is not defined for option 'gpu-architecture'
Makefile:75: recipe for target 'cuda_memtest.o' failed
make: *** [cuda_memtest.o] Error 1

and when I adjusted Makefile to inlcude sm_30 (what cuda 10 allows)

```
$/usr/local/tmp/cuda_memtest-dev $ make
nvcc -c -arch sm_50 -DSM_50 -O3  -I. -I/usr/local/cuda/include  -o cuda_memtest.o cuda_memtest.cu
cuda_memtest.cu(381): error: identifier "ENABLE_NVML" is undefined

1 error detected in the compilation of "/tmp/tmpxft_0000d7c5_00000000-6_cuda_memtest.cpp1.ii".
Makefile:75: recipe for target 'cuda_memtest.o' failed
make: *** [cuda_memtest.o] Error 1
```

have no idea what to do next.

I have noticed that with minimization mdp file all goes well with no problem
i.e.

```
integrator      = steep or sd
```

best,

Dmytro

#11 - 11/22/2018 05:11 PM - Szilárd Páll

I used cmake to build memtest and had no issues. You have ECC disabled, but if you had it enable we might be able to see errors that the card itself may report, so you could try enabling ECC, trying to reproduce the error, and inspecting the nvidia-smi log's relevant section.

#12 - 01/18/2019 11:56 AM - Szilárd Páll

- Status changed from New to Feedback wanted

Does the issue still persist with the latest 2018 and/or 2019 releases?

#13 - 01/30/2019 06:17 PM - Mark Abraham

Perhaps related to report at https://mailman-1.sys.kth.se/pipermail/gromacs.org_gmx-users/2019-January/124029.html

#14 - 03/15/2019 04:00 PM - Szilárd Páll

Mark Abraham wrote:

> Perhaps related to report at https://mailman-1.sys.kth.se/pipermail/gromacs.org_gmx-users/2019-January/124029.html

It might be, but we'd need feedback from OP to be able to determine.

#15 - 04/09/2019 09:05 PM - Szilárd Páll

I am starting to see more similarities between this issue and the gmx-users report I am trying to debug:
- both use Intel Skylake Gold 6xxx CPUs
- both use AVX512 builds
- both use GP102 Quadro GPUs

Dmytro, can you help us test some assumptions? First off: if you build with GMX_SIMD=AVX2_256 does the error still reproduce?

#16 - 04/11/2019 04:30 PM - Szilárd Páll

- Subject changed from GPU isue with CUDA 10 to GPU isue with Ubuntu 18.04 / glibc 2.27 (?) and CUDA 9.2/10

- Status changed from Feedback wanted to In Progress

- Target version set to 2019.2

Based on the feedback from Jon Vincent on the users' list, this *seems* to be a compatibility issue between some at the least the Ubuntu 18.04 + glibc 2.27 and CUDA 9.2 / 10 combination, possibly affecting other similar combinations too.

Depending on how confident we are in what the issue is and how it manifests we should:

- if our knowledge of the problem remains as vague as it is, we should add an item to the "known issues" section of the docs
- if the issue can be well identified we should add more clear notes in the docs and a warning note in cmake too.

#17 - 04/11/2019 04:57 PM - Jonathan Vincent

Note this is not specifically related to CUDA. I have seen similar problems (e.g. Gromacs crashing) when Gromacs is built without GPU support at all on Ubuntu 18.04 machines and run with 20 or so threads per MPI rank. Potentially seen more on CUDA runs as they will be generally run with more threads.

What I have seen

- Problem with Ubuntu 18.04 LTS using the default compiler (7.3) and default glibc
- Using an earlier glibc (e.g. by building gcc 7.3 on a RHEL 6 box) does not have the same problem
- CPU only runs (using a binary built without GPU support at all) shows similar problems when built using Ubuntu 18.04 LTS defaults.
- Jobs fail with large numbers of threads (approx 20 per rank), but seem to work fine with lower numbers on both CPU and GPU

Building gcc 8.x and using that fixes the problem on CPU and GPU. CUDA 10.1 is required for gcc 8 support with CUDA.

#18 - 04/11/2019 09:36 PM - Szilárd Páll

Jonathan Vincent wrote:

> Note this is not specifically related to CUDA. I have seen similar problems (e.g. Gromacs crashing) when Gromacs is built without GPU support at all on Ubuntu 18.04 machines and run with 20 or so threads per MPI rank. Potentially seen more on CUDA runs as they will be generally run with more threads.

That is plausible, but we need logs/configs/errors of those instances because so far we have only seen the issue in CUDA runs. I have certainly ran non-GPU regressiontests with >20 threads.

> Building gcc 8.x and using that fixes the problem on CPU and GPU. CUDA 10.1 is required for gcc 8 support with CUDA.

We need to translate this to action/code in a release, but we can't require users to compile a gcc (and install new CUDA if relevant) unless we have a solid description/repro of the cases where this issue appears. So far we have CUDA-only repro cases, so that's the only things we *could* claim, but I'd rather not have vague claims in release notes/docs (let alone warning against or even blocking some compiler/glibc combinations in cmake).

#19 - 04/15/2019 04:35 PM - Paul Bauer

- Target version changed from 2019.2 to 2019.3

the release is today and there has been no concrete information for us to work on, so I'll be bumping this to the next point release

#20 - 04/24/2019 01:07 PM - Szilárd Páll

We need some more feedback on this, Jon can you get back with repro cases?

#21 - 05/06/2019 07:10 PM - Szilárd Páll

- Difficulty hard added

- Difficulty deleted (uncategorized)

I have reproduced the issue with regressiontests failing when >20 OpenMP threads are used under the following circumstances:

- gcc 7.3 (both vanilla and Ubuntu version);
- glibc 2.27 *shipped by Ubuntu 18.04*
- a build with -mavx512f; default with -DGMX_SIMD=AVX_512, but adding the flag to any other build also reproduces the issue.
- compiler involvement is clearly needed as doing the compile on a host with glibc 2.23, then doing only the final linking against 2.27 eliminates the issue.

This indicates that it is a glibc 2.27 + gcc 7.3 + AVX512 + OpenMP bug, but not sure whether in the compiler, gomp or glibc.

What is not known yet:

- if a vanilla glibc 2.27 eliminates the issue
- if glibc >2.27 eliminates the issue
- if gcc 7.4 solves the issue
- where exactly is the bug originating from (which source) -- this could point to a potential workaround

It would be good to test the above questions to know a bit more and be able to address the matter better. For the latter point, we should script adding the -mavx512f to every source file (that has OpenMP?) one by one and check when the bug is triggered.

#22 - 05/06/2019 07:13 PM - Szilárd Páll

- Target version changed from 2019.3 to 2018.7

- Affected version - extra info set to 2019.x

Retargeting this for 2018. Contributions are needed, the minimum is some "known issues" text, but ideally we'd want to know more and attempt to have a workaround. I will however not have more time to dig into this before 2018.7.

#23 - 05/08/2019 12:03 PM - Szilárd Páll

- Subject changed from GPU isue with Ubuntu 18.04 / glibc 2.27 (?) and CUDA 9.2/10 to incorrect results with Ubuntu 18.04 / glibc 2.27 (?) and >20

threads

#24 - 05/14/2019 10:43 AM - Paul Bauer

while trying to reproduce this I found this in a RelWithAssert build, not sure if you already saw it as well, Szilard

```
Assertion failed:
Condition: ncjTotalNew == ncjTotal
The total size of the lists before and after rebalancing should match
```

Happens when running >=20 threads and AVX512, not in the other cases as observed before, and consistent between 2018.6 and 2019.2 build from the tarball

#25 - 05/14/2019 11:45 AM - Berk Hess

Is this with the input attached here?
Is that with nb on cpu or gpu?

#26 - 05/14/2019 11:46 AM - Paul Bauer

this is with the quick.tpr Mark attached.
I ran on dev-purley01 with this command

```
gmx mdrun -s quick.tpr -ntmpi 1 -ntomp 20
```

#27 - 05/15/2019 11:28 AM - Paul Bauer

- File mdrun.out added

- File md.log added

Attached log files for sample run

#28 - 05/15/2019 02:23 PM - Berk Hess

The assertion only fails with a release with assert build. Things seem to work fine with release with debug info. So the issue seems to be related to some optimization. But then I don't know how to debug this.
I printed the counts that go into the failing assertion and 42 out of 217678 cluster pairs seem to be missing.

#29 - 05/15/2019 03:51 PM - Berk Hess

With non-bondeds on the CPUs, not calling rebalanceSimpleLists() (where the assertion fails) gives correct results for the test case.

There is not much code in rebalanceSimpleLists() and it looks rather simple, so I have no idea yet which part could cause the issue or be affected by a compiler bug.

#30 - 05/15/2019 04:28 PM - Berk Hess

The issue is somewhere else in the search code. With release with assert the number of cluster pairs is a factor 5 smaller than it should be.

#31 - 05/16/2019 09:36 AM - Paul Bauer

I checked builds both with -O3 and -O2, with the crash only happening with -O3. I could try to see which optimization (or combination thereof) triggers it in the end.

#32 - 05/16/2019 10:41 AM - Berk Hess

The issue is that the results of a one line loop summing ints over a list of objects through pointers gives only the sum over the first 4 elements instead of all 20.
Adding a GMX_RELEASE_ASSERT in the loop body avoids the issue. So this looks like a loop unrolling bug.

#33 - 05/16/2019 10:47 AM - Paul Bauer

so, I checked some more and changed the RelWithDebugInfo flags from

```
-O2 -g -DNDEBUG
```

to

```
-O3 -g -DNDEBUG
```

and got a segmentation fault instead when running with -ntomp 20

```
There are: 78646 Atoms

Started mdrun on rank 0 Thu May 16 10:42:03 2019
           Step           Time
              0        0.00000

   Energies (kJ/mol)
          Angle    Proper Dih.  Improper Dih.          LJ-14     Coulomb-14
    1.67779e+04    2.36909e+04    9.57938e+02    8.54109e+03    1.06861e+05
        LJ (SR)  Disper. corr.   Coulomb (SR)   Coul. recip.      Potential
    4.83611e+04   -1.54691e+04   -3.07971e+05    1.14580e+04   -1.06792e+05
    Kinetic En.   Total Energy  Conserved En.    Temperature Pres. DC (bar)
    1.97639e+05    9.08475e+04    9.08943e+04    3.02025e+02   -3.31363e+02
 Pressure (bar)   Constr. rmsd
    7.70312e+02    3.46855e-05

Using 1 MPI thread
Using 20 OpenMP threads 

NOTE: The number of threads is not equal to the number of (logical) cores
      and the -pin option is set to auto: will not pin threads to cores.
      This can lead to significant performance degradation.
      Consider using -pin on (and -pinoffset in case you run multiple jobs).

Back Off! I just backed up traj_comp.xtc to ./#traj_comp.xtc.4#

Back Off! I just backed up ener.edr to ./#ener.edr.4#
starting mdrun '2 SYMMETRICAL 'FOOTBALL' COMPLEX in water'
10000 steps,     20.0 ps.

step 17: One or more water molecules can not be settled.
Check for bad contacts and/or reduce the timestep if appropriate.
Wrote pdb files with previous and current coordinates
```

#34 - 05/16/2019 11:27 AM - Berk Hess

I saw now exactly what goes wrong. The result of the sum over 20 is only over 0, 1, 18, 19. So the 16 middle elements are missing. This looks like AVX512 vectorization adding only the start and end parts and not the main loop part.
With 24 threads I see 0-16 missing.

#35 - 05/16/2019 11:50 AM - Erik Lindahl

I think the key question is whether the array data is always correct and the error only occurs in the summation loop, or if we happen to overwrite 16 elements of memory semi-randomly with zeros.

In the second case, adding either printf or assertion statements might alter the memory access patterns and hide the bug, so before concluding this is a compiler/library bug we are just working around we probably need to be able to reproduce it inside a debugger and be able to inspect what is happening (and conversely, inspect the assembly to confirm it's incorrect).

#36 - 05/16/2019 11:51 AM - Paul Bauer

So, spending way too much time on trying optimization flags, I found that

```
-ftree-loop-vectorize
```

triggers the bug when added to on -O2 build.

#37 - 05/16/2019 11:52 AM - Paul Bauer

This is what it does according to the gcc manual

```
-ftree-loop-vectorize

    Perform loop vectorization on trees. This flag is enabled by default at -O3 and by -ftree-vectorize, -fpro
file-use, and -fauto-profile.
```

#38 - 05/16/2019 01:52 PM - Berk Hess

- File nbnxn_sum.cpp added

- File nbnxn_sum.s added

The bug is still triggered with this loop in a separate function in a separate source file.
I generated assembly by adding -S -fverbose-asm in cmake, but that wrtites the assembly to the .o files, so I can't actually check that this code would also give the error. But as no other flags changes, I would assume so.
Code and assembly attached.

#39 - 05/16/2019 02:39 PM - Berk Hess

- File nbnxn_sum.s.avx2_256 added

Here is also the assembly for avx2_256, which should be correct. The global vectorization setup seems to be rather similar, but, as expected, many of the instructions differ.

#40 - 05/16/2019 09:39 PM - Berk Hess

I want to check gcc 5 and 6, but the build on our purley machine complains:
/opt/tcbsys/gcc/5.2/lib64/libstdc++.so.6: version `CXXABI_1.3.11' not found (required by bin/gmx)

#41 - 05/20/2019 10:15 AM - Paul Bauer

- File nbnxn_search-nobug.asm added

- File nbnxn_search-bug.asm added

For completeness I added the asm files for both the build with a fix, and one where the pragma has been commented out.
I tried reading the files but can't say much, but it seems like the loop is completely optimized away in the case of the bug.
Both builds where done with -O3 -g -S -fverbose-asm.

#42 - 05/20/2019 07:33 PM - Szilárd Páll

Berk Hess wrote:

> The bug is still triggered with this loop in a separate function in a separate source file.
> I generated assembly by adding -S -fverbose-asm in cmake, but that wrtites the assembly to the .o files, so I can't actually check that this code would also give the error. But as no other flags changes, I would assume so.
> Code and assembly attached.

Do you have a patch file for the 2019 source or the preprocessed cpp source? I wanted to check the output of goldbolt and see if there is still an issue even in gcc 7.4 with optimizations on.

#43 - 05/20/2019 07:47 PM - Szilárd Páll

Szilárd Páll wrote:

> Berk Hess wrote:
>
>> The bug is still triggered with this loop in a separate function in a separate source file.
>> I generated assembly by adding -S -fverbose-asm in cmake, but that wrtites the assembly to the .o files, so I can't actually check that this code would also give the error. But as no other flags changes, I would assume so.
>> Code and assembly attached.
>
> Do you have a patch file for the 2019 source or the preprocessed cpp source? I wanted to check the output of goldbolt and see if there is still an issue even in gcc 7.4 with optimizations on.

Made a preprocessed source myself. I see no difference between gcc 7.3 and 7.3 and even 7.2 seems to only differ in slight register allocation.
https://godbolt.org/z/J9aaU5

#44 - 05/21/2019 01:59 PM - Berk Hess

- File nbnxn_sum.cpp.o added

Here is my assembly with gcc 7.4.0 from OpenSuse. It looks the same as that from gcc 6, but with the extra .long lines for avx512.

#45 - 05/21/2019 03:16 PM - Szilárd Páll

Berk Hess wrote:

> Here is my assembly with gcc 7.4.0 from OpenSuse. It looks the same as that from gcc 6, but with the extra .long lines for avx512.

I copied the exact same compiler flags into godbolt, but I can't reproduce it:
https://godbolt.org/z/pgUjge

Any ideas? Perhaps distros backported something and vanilla gcc doesn't have the fix?

#46 - 05/21/2019 05:05 PM - Paul Bauer

I went through the gcc release notes and bugs and didn't see anything that was directly related to our issue here. There was only mentioned issue with the optimization option and avx512, but in the mentioned bug this lead to an ICE instead.

#47 - 05/27/2019 02:44 PM - Mark Abraham

- Status changed from In Progress to Fix uploaded

#48 - 05/29/2019 08:15 AM - Berk Hess

- Status changed from Fix uploaded to Resolved

Applied in changeset 9f45a4be4b5e24f49c4c1ce8db8144b21631a891.

#49 - 06/03/2019 01:05 PM - Paul Bauer

- Status changed from Resolved to Closed

#50 - 06/30/2019 07:50 PM - Dmytro Kovalskyy

Hi, I am sorry for getting off this thread.

I just updated gromacs and here is what I found.

With 2018-7 release the bug is still persistent, if I use following compiling options
cmake .. -DGMX_GPU=ON -DGMX_USE_NVML=ON

but if I compile with AVX activated the bug is gone
cmake .. -DGMX_GPU=ON -DGMX_USE_NVML=ON -DGMX_SIMD=AVX2_256

At least it now works with my P5000 card.
Thank you a lot!

#51 - 07/01/2019 11:26 PM - Szilárd Páll

- Status changed from Closed to Blocked, need info

- Target version deleted (2018.7)

Dmytro, thanks for testing; quite unexpectedly, the workaround that does solve the issue in the 2019 branch does not seem to have the intended effect in the 2018.7 (at least I am getting a segv in the nbnxn-free-energy and nbnxn-free-energy-vv regressiontets with 20 threads). We'll look into this, but as the 2018 is officially out of maintenance, we might not do a new release.

However, rest assured that in your setup you are not loosing performance, in fact in GPU accelerated runs AVX2_256 is generally faster even on Skylake Gold CPUs.

Files

md200ns.tpr	3.98 MB	11/15/2018	Dmytro Kovalskyy
md200ns.log	17.8 KB	11/15/2018	Dmytro Kovalskyy
quick.out	7.71 KB	11/16/2018	Mark Abraham
quick.log	29.2 KB	11/16/2018	Mark Abraham
quick.tpr	3.98 MB	11/16/2018	Mark Abraham
mdrun.out	2.89 KB	05/15/2019	Paul Bauer
md.log	17 KB	05/15/2019	Paul Bauer
nbnxn_sum.cpp	2.69 KB	05/16/2019	Berk Hess
nbnxn_sum.s	30.6 KB	05/16/2019	Berk Hess
nbnxn_sum.s.avx2_256	27.8 KB	05/16/2019	Berk Hess
nbnxn_search-nobug.asm	1.49 MB	05/20/2019	Paul Bauer
nbnxn_search-bug.asm	1.53 MB	05/20/2019	Paul Bauer
nbnxn_sum.cpp.o	7.62 KB	05/21/2019	Berk Hess