GROMACS - Bug #2007

Incorrect forces with LJ potential-switch, GPU and PME tuning

07/15/2016 10:51 AM - Yannic Alber

Status:	Closed		
Priority:	High		
Assignee:	Berk Hess		
Category:	mdrun		
Target version:	5.1.4		
Affected version - extra info:	2016-rc1	**Difficulty:**	
Affected version:	5.1.2		

Description

Dear all,

we struggle to get a TI on our computer running. The specifications are listed below. As you can see, its a two socket, two graphics cards machine. Therefore, the plan is to run two simulations in parallel. But we can't get a single one to run.

```
Running on 1 node with total 20 cores, 20 logical cores, 2 compatible GPUs
Hardware detected:
  CPU info:
    Vendor: GenuineIntel
    Brand:  Intel(R) Xeon(R) CPU E5-2640 v4  <at>  2.40GHz
    SIMD instructions most likely to fit this hardware: AVX2_256
    SIMD instructions selected at GROMACS compile time: AVX2_256
  GPU info:
    Number of GPUs detected: 2
    #0: NVIDIA GeForce GTX 1080, compute cap.: 6.1, ECC:  no, stat:
compatible
    #1: NVIDIA GeForce GTX 1080, compute cap.: 6.1, ECC:  no, stat:
compatible
```

The simulation system in question is a protein-ligand-complex in TIP3P-water and amber ff99SB as force field.

Now lets get into the messy details. We tried different mdrun commandline argument rotations, for example:

```
gmx mdrun -s md.tpr -pin on -ntomp 2 -ntmpi 5 -gpu_id 00000 -deffnm md
(does not work)

gmx mdrun -s md.tpr -pin on -ntomp 5 -ntmpi 2 -gpu_id 00 -deffnm md
```

Associated revisions

Revision 2fffbdbf - 08/09/2016 02:18 PM - Berk Hess

Fix twin-cutoff for GPU LJ pot-switch

With rcoulomb>rvdw, as used with PME tuning, the GPU kernels performed the LJ cut-off check before applying the potential-switch to the LJ forces and energies.

Fixes #2007.

Change-Id: Ie2f6fdfec10bdb8800123e6fa299839a31bc0f93

History

#1 - 07/15/2016 10:28 PM - Mark Abraham

- Status changed from New to Accepted

- Affected version - extra info set to 2016-rc1

Thanks for the report. I have reproduced the tpr blowing up on tcbs23 (2 GPU, 32 cores) within a few hundred MD steps when run with mdrun defaults (5.1, 5.1.2 and 2016-rc1).

With mdrun -nb cpu it runs for > 1500 steps.

With mdrun -notunepme it runs for > 17000 steps even with GPUs, so we can probably exclude problems with CUDA 8 or latest-model GPUs. This probably gives Yannic a way to move forward, even though throughput will be inferior (because the auto-tuning is disabled), until we work out what is going wrong.

With mdrun -dlb no -tunepme it also crashes.

Yannic, can you please also upload a tarball of the inputs you used with grompp, so we can make a .tpr that runs with version 5, to see if the issue is present there also? (Or make one yourself if you prefer to do that, but the tarball will give us more extensive options.)

#2 - 07/15/2016 10:29 PM - Mark Abraham

- Description updated

#3 - 08/03/2016 11:03 PM - Szilárd Páll

Could we have the input files, please? Something is indeed wrong here, but tracing down the issue could be easier if we had the gro/top/mdp files.

#4 - 08/08/2016 04:57 PM - Berk Hess

- Status changed from Accepted to In Progress

- Assignee set to Berk Hess

The issue is also present in release 2016.
If I use -notunepme the issue does not occur.
I tried calculating and printing energies to see where the issue arises, but setting both to 1 or 2 steps makes the issue go away. This seems to indicate it is some kind of timing issue.
Adding -pforce 10000 shows large forces at step 162, just after the second PME grid setting:
step 80: timed with pme grid 108 108 108, coulomb cutoff 1.200: 11485.6 M-cycles
step 160: timed with pme grid 100 100 100, coulomb cutoff 1.251: 8075.2 M-cycles

The large forces don't seem to come from the free-energy kernels, so I guess the issue is on the GPU, but I'm not sure about that.

PS rlist=1.8 gives a buffer of 0.6 nm and an enormous increase in calculation cost. For production I would suggest to use a low verlet-buffer tolerance value instead.

#5 - 08/09/2016 11:12 AM - Berk Hess

- Subject changed from Problems with TI on GPUs to Sudden incorrect forces with CUDA and PME tuning

- Priority changed from Normal to High

I have now reproduced this issue with v2016 without free-energy and with -ntmpi 1.
After the second PME grid change at step 160 things go wrong. At steps 160 and 161 the force rmsd CPU vs GPU is 4 (which indicates fully correct forces), at step 162 it is 300. So the strange thing is that it only seems to go wrong at the third step with the second grid change. It doesn't always go wrong, but if it does it always happens at step 162.

#6 - 08/09/2016 12:43 PM - Berk Hess

- Subject changed from Sudden incorrect forces with CUDA and PME tuning to Incorrect forces with LJ potential-switch, GPU and PME tuning

In the GPU kernels, CUDA and OpenCL, the LJ cut-off masking is done before calculating the LJ forces. This means the forces will be incorrect when a twin-range cut-off is used, as happens with PME tuning.

#7 - 08/09/2016 01:31 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2007.
Uploader: Berk Hess (hess@kth.se)
Change-Id: Ie2f6fdfec10bdb8800123e6fa299839a31bc0f93
Gerrit URL: https://gerrit.gromacs.org/6106

#8 - 08/09/2016 01:47 PM - Berk Hess

- Status changed from In Progress to Fix uploaded

- Target version changed from 2016 to 5.1.4

#9 - 08/15/2016 08:04 PM - Berk Hess

- Status changed from Fix uploaded to Resolved

Applied in changeset 2fffbdbf8dd8573714661f4f6eeb46221a40f094.

#10 - 09/07/2016 02:04 PM - Mark Abraham

- Status changed from Resolved to Closed

Files

md1_ntomp2_ntmpi5_gpu00000.tpr	4.3 MB	07/15/2016	Yannic Alber
md1_ntomp2_ntmpi5_gpu00000.log	21.3 KB	07/15/2016	Yannic Alber